

Study Material

“COMPUTER SCIENCE”

Class - XII

CONTENTS

Topic	Page No
Unit 1: Programming in C++	3-75
C++ Revision Tour	3-25
Object Oriented Programming	26-27
Function Overloading	28-29
Classes And Objects	30-35
Constructors And Destructors	36-42
Inheritance: Extending Classes	43-52
Data File Handling	53-67
Pointers	68-75
Unit 2: Data Structures	76-100
2.1 Arrays	76-86
2.2 Stack	87-95
2.3 Queue	96-100
Unit 3: Database and SQL	101-107
Database Concepts	101-102
Structured Query Language	103-107

Unit 4: Boolean Algebra	108-118
Unit 5: Communication & Open Source Concepts	119-127
HOTS QUESTIONS	128-138
Sample Paper	139-147
Practice Paper	148-154

Unit 1: Programming in C++

: C++ Revision Tour

REVISION

History, C++ Character sets, C++ Tokens, #define directive, Type conversion, sizeof() operator, typedef keyword, Arrays

History:

- Developed at AT&T Laboratory in the early 1980" s by Bjarne Stroustrup .
- Also known as "C with Classes".
- Supports Object oriented technology to develop software, most near to the real world.

C++ Character Sets:

Letters	A-Z , a-z
Digits	0-9
Special Symbols	Space + - * / ^ \ () [] { } = ! = < > . , " \$, ; : % ! & ? _ # < = > = @
White Spaces	Blank spaces, horizontal tab, carriage return
Other Characters	Any of the 256 ASCII character

C++ Tokens:

The smallest individual unit of the program is known as Token.

Keywords	Words reserved by the language with a special meaning. Like int, float, switch, case etc
Identifiers	Long sequence of letters & digits to identify a memory block, program unit or program objects. Like name, a, x, A, X, date, file1, file2 etc.
Literals	Constants that never changes their value during the execution of a program. Type : <ul style="list-style-type: none"> • Integer Constant <ul style="list-style-type: none"> ➤ Decimal - 1234, +97 (not begins with zero) ➤ Octal - 014, 023 (preceded by 0) ➤ Hexadecimal - 0XC (preceded by 0x or 0X) • Character Constant („z“ , „A“ , „\a“ , „\t“ , „\n“ , „0“ , „3“) • Floating Constant (-13.0, -0.00065, 1.52E07, 0.172E-3) • String Constants (“abc\0”, “Computer Science\0”)
Punctuators	[] () { } , ; : * = #
Operators	<<, >>, +, -, *, /, %, ++, --, ==, <, >, <=, >=, !=, &&, , !

#define directive (Defined constants)

We can define our own names for constants that we use very often without having to resort to memory-consuming variables, simply by using the #define preprocessor directive. Its format is:

```
#define identifier value
```

For example:

```

#include<iostream.h>
#define PI 3.14159      // This defines two new constants: PI and
NEWLINE. #define NEWLINE '\n'    // Once they are defined, they can be
used

                                // in the code in place of their values

int main ()
{
    double r=5.0;                // radius
    double circle;

    circle = 2 * PI * r;
    cout << circle;
    cout << NEWLINE;

    return 0;
}

```

Output :

31.4159

In fact the only thing that the compiler preprocessor does when it encounters #define directives is to literally replace any occurrence of their identifier (in the previous example, these were *PI* and *NEWLINE*) by the code to which they have been defined (31.4159 and „\n” respectively). The #define directive is not a C++ statement but a directive for the preprocessor; therefore it assumes the entire line as the directive and does not require a semicolon (;) at its end. If we append a semicolon character (;) at the end, it will also be appended in all occurrences of the identifier within the body of the program that the preprocessor replaces.

Declared constants (const)

With the const prefix we can declare constants with a specific type in the same way as we would do with a variable. For example:

```

const int   pathwidth = 100;
const char  tabulator = '\t';

```

Here, pathwidth and tabulator are two typed constants. They are treated just like regular variables except that their values cannot be modified after their definition.

Type Conversion in C++:

1. Implicit type conversion
2. Explicit type conversion (type casting)

1. **Implicit conversion (Type promotion):** Implicit conversions do not require any operator. They are automatically performed when a value is copied to a compatible type. Usual arithmetic implicit type conversions are summarized in the following table:

Step No.	If either type of operand is	Then resultant type of expression is	Otherwise
1	long double	long double	step 2
2	Double	double	step 3
3	Float	float	step 4
4	short int or int(signed or unsigned) or char	int	step 5
5	unsigned long	unsigned long	step 6
6	long int or unsigned int	(i) long int (provided long int can represent all values of unsigned int) (ii) unsigned long int (if all values of	step 7

		unsigned int can't be represented by long int)	
7	Long	long	step 8
8	Unsigned	unsigned	both operands are int

- 2. Explicit type conversion (Type Casting):** Type casting operators allow us to convert a datum of a given type to another. There are several ways to do this in C++. The simplest one, which has been inherited from the C language, is to precede the expression to be converted by the new type enclosed between parentheses ():

```
int i;
float f = 3.14;
i = (int) f;
```

The previous code converts the float number 3.14 to an integer value (3), the remainder is lost. Here, the typecasting operator was (int). Another way to do the same thing in C++ is using the functional notation: preceding the expression to be converted by the type and enclosing the expression between parentheses:

```
i = int ( f );
```

Both ways of type casting are valid in C++.

sizeof() operator:

This operator accepts one parameter, which can be either a type or a variable itself and returns the size in bytes of that type or object:

```
a = sizeof (char);
```

This will assign the value 1 to „a“ because char is a one-byte long type. The value returned by sizeof is a constant, so it is always determined before program execution.

typedef keyword:

Using the keyword typedef, we can create an alias (a synonym) for existing fundamental or compound datatypes in C++. **Syntax:**

```
typedef existing_type new_type_name ;
```

where existing_type is a C++ fundamental or compound type and new_type_name is the name for the new type we are defining. For example:

```
typedef char C;
typedef unsigned int WORD;
typedef char * pChar;
typedef char field [50];
```

In this case we have defined four data types: C, WORD, pChar and field as char, unsigned int, char* and char[50] respectively, that we could perfectly use in declarations later as any other valid type:

```
C mychar, anotherchar, *ptc1;
WORD myword;
pChar ptc2;
field name;
```

typedef does not create different types. It only creates synonyms of existing types. That means that the type of myword can be considered to be either WORD or unsigned int, since both are in fact the same type.

Benefit of using typedef

- typedef can be useful to define an alias for a type that is frequently used within a program.
- It is also useful to define types when it is possible that we will need to change the type in later versions of our program, or if a type we want to use has a name that is too long or confusing.

UNSOLVED PROBLEMS

- Problem 1: Write two advantages of using include compiler directive.
- Problem 2: What is the difference between type casting and automatic type conversion? Explain with suitable example.
- Problem 3: What is the purpose of using a **typedef** command in C++? Explain with suitable example.
- Problem 4: What is the difference between **#define** preprocessor directive and **const** in C++? Explain with suitable example.

ARRAYS

Introduction to Arrays, Types of Array, One Dimensional Array, Two Dimensional Array

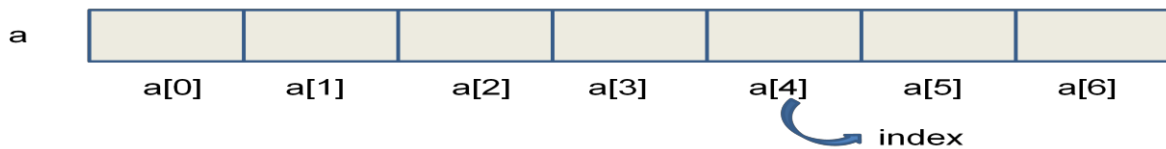
Introduction to Arrays:

Collection of variables of same data type that are referenced by a common name. Example:

Scalar Variables in memory



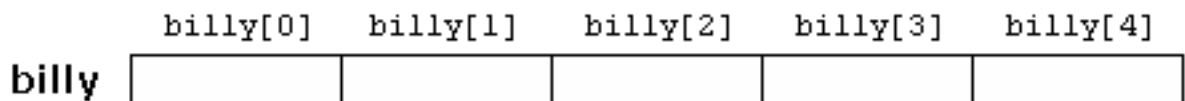
Array in Memory



Types of Array:

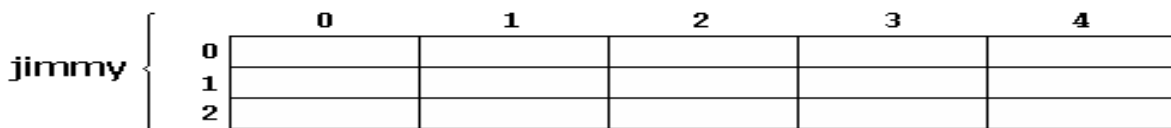
- Single or One Dimensional

```
int billy[5];
```



- Double or Two Dimensional

```
int jimmy [3][5];
```



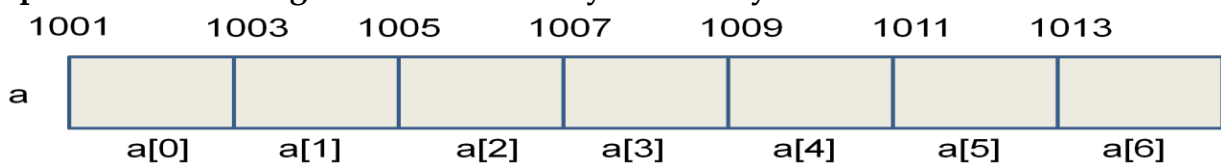
One Dimensional Array:

Declaration

```
datatype name[No.of elements]={initial values of elements separated by comma};
```

Example - `int a [7] = { 16, 2, 77, 40, 120,7,1 };`

Representation of single dimensional array in memory:



Points to remember:

Base Type of Array : int

(The data type of array is known as its base type)

Base Address : 1001

(The address of the very first element of the array)

Size in bytes = Size of base type * No. of elements

Example:

```

/*Program to accept 10 numbers and display them along with their sum
after storing in an array.*/
#include <iostream.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int a[10],s=0;
    cout<<"Enter 10 numbers : ";
    for(int i=0;i<10;i++) //Taking input
        cin>>a[i];
    for(i=0;i<10;i++) //Processing array by reading elements
one
        s=s+a[i]; // by one & adding them in a variable s
    for(i=0;i<10;i++) //Displaying output
        cout<<a[i]<<"\n ";
    cout<<"Sum : "<<s;
    getch( );
}

```

Passing Array to a function

```

#include <iostream.h>
#include<conio.h>
void main( )
{
    clrscr( );
    int sum(int a[]);
    int a[10],s=0;
    cout<<"Enter 10 numbers : ";
    for(int i=0;i<10;i++)
        cin>>a[i];
    s=sum(a);
    for(i=0;i<10;i++)
        cout<<a[i]<<"\n ";
    cout<<"Sum : "<<s;
    getch( );
}

```

```

int sum(int a[] )
{
    int res=0;
    for(int i=0;i<10;i++)
        res=res+a[i];
    return res;
}

```

```

int sum(int a[10] )
{
    int res=0;
    for(int i=0;i<10;i++)
        res=res+a[i];
    return res;
}

```

```

int sum(int *a )
{
    int res=0;
    for(int i=0;i<10;i++)
        res=res+a[i];
    return res;
}

```

Returning Array from a function

```

#include <iostream.h>
#include<conio.h>
void main( )
{
    clrscr( );
    void sort(int *a);
    int a[10];
    cout<<"Enter 10 numbers :";
    for(int i=0;i<10;i++)
        cin>>a[i];
    sort(a);
    for(i=0;i<10;i++)
        cout<<a[i]<<"\n";
    getch( );
}

```

```

void sort(int *a )
{
    int t;
    for(int i=0;i<10;i++)
    for(int j=0;j<10;j++)
    {
        if(a[i]<a[j])
        {
            t=a[i];
            a[i]=a[j];
            a[j]=t;
        }
    }
}

```

Note: Array is always implicitly passed by reference, therefore no need to return it, the changes automatically reflected back to actual array.

Character Sequences as single dimensional array:

(String as an array of Characters)

Example:

```
char a [8] = "Class12";
or char a [8] = {"C", "l", "a", "s", "s", "1", "2", "\0"};
```

1001	1002	1003	1004	1005	1006	1007	1008
C	l	a	s	s	1	2	\0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]

Example :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    char a[8]="Class12";
    clrscr();
    cout<<a;
    getch();
}
```

Passing Character array to a function

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    char a[8]="Class12\0";
    void disp(char *);
    clrscr();
    disp(a);
    getch();
}
```

```
void disp(char b[])
{
    puts(b);
}
```

Two dimensional Array:

Declaration:

```
datatype    name [Row] [Column];
```

Example: `int a[3] [7] ;`

Representation of two dimensional array in memory:

	1001	1003	1005	1007	1009	1011	1013
a	a[0][0]	a[0][1]					a[0][6]
	a[1][0]	a[1][1]					a[1][6]
	a[2][0]	a[2][1]					a[2][6]

Points to remember:

Base Type of Array	:	int
Base Address	:	1001
(The address of the very first element of the array)		

UNSOLVED PROBLEMS

- Problem 1: Rewrite the following program after removing the syntactical error(s), if any. Underline each correction.
- ```
#include <iostream.h>
const int Size 5;
void main()
{int Array [Size] ;
 Array = { 50, 40, 30, 20, 10 } ;
 for (Ctr = 0 ; Ctr < Size; Ctr++)
 cout>>Array [Ctr];
}
```
- Problem 2: Rewrite the following program after removing all the syntax error(s), if any. Underline each correction.
- (i) #include (iostream.h)
- ```
void main( )
{ int X[ ] = { 60,50,30,40}, Y; Count = 4;
  cin >> Y ;
  for ( I = Count - 1 ; I >= 0, I - -)
    switch( I )
    {case 0 :
      case 2 : cout<<Y * X [I ] << endl; break;
      case 1 :
      case 3 : cout >> Y + X [I] ;
    }
}
```
- (ii) #include <iostream.h>
- ```
void main ()
{int P[]={90,10,24,15}:Q, Number = 4 ; Q = 9;
 for [int I = Number - 1 ; I >= 0, I--]
 Switch (I)
 {case 0 :
 case 3 : cout >> P [I] * Q << endl ; break ;
 case 1 :
 case 2 : cout << P [I] + Q ;
 }
}
```
- Problem 3: Write a C++ program to find the sum of two matrices..
- Problem 4: Write a C++ program to find the difference of two matrices.
- Problem 5: Write a C++ program to find the product of two matrices.
- Problem 6: Write a C++ program to find the transpose of a m X n matrix.
- Problem 7: Write a C++ program to find the sum of the diagonal elements of 4X4 matrix.

## **FUNCTIONS**

Functions in C++, Benefits, Types of Functions in C++, Defining Functions & returning values from functions, Scope of Variables, Parameters and their type (Actual & Formal Parameters), Calling Functions, Call by value, Call by reference

### **Functions in C++:**

A named set of statements (subprogram) that may take some values from its caller and returns a value after performing a specific job.

### **Benefits of using functions:**

- Provides modularity.
- Provides reusability.
- Makes program modification easier.
- Makes program smaller and simpler.

### **Types of Functions in C++**

Two types of functions are available in C++.

1. **Library functions:** Provided by C++ compiler in library and can be used by including respective header file.
2. **User defined functions:** Defined by user.

### **Defining Functions & returning values from functions.**

A function is a group of statements that is executed when it is called from some point of the program. The following is its format:

```
<return type> <function name> (parameter1, parameter2, ...) { statements
}
```

where:

- return type is the data type specifier of the data returned by the function. If function does not returns any value return type should be void.
- name is the identifier by which it will be possible to call the function.
- parameters (as many as needed): Each parameter consists of a data type specifier followed by an identifier, like any regular variable declaration (for example: int x) and which acts within the function as a regular local variable. They allow to pass arguments to the function when it is called. The different parameters are separated by commas.
- statements is the function's body. It is a block of statements surrounded by braces {}.

### **Example:**

```
#include <iostream.h>
int addition (int a, int b)
{
 int r;
 r=a+b;
 return (r);
}
void main ()
{ int z;
 z = addition (5,3);
 cout << "The result is " << z;
}
```

### **Output :**

The result is 8

**Explanation:** A C++ program always begins its execution by the main function. So we will begin there. We can see how the main function begins by declaring the variable z of type int. Right after that, we see a call to a function called addition.

```
int addition (int a, int b)
z = addition (5 , 3);
```

The parameters and arguments have a clear correspondence. Within the main function we called to addition passing two values: 5 and 3, that correspond to the int a and int b parameters declared for function addition.

At the point at which the function is called from within main, the control is lost by main and passed to function addition. The value of both arguments passed in the call (5 and 3) are copied to the local variables int a and int b within the function.

Function addition declares another local variable (int r), and by means of the expression r=a+b, it assigns to r the result of a plus b. Because the actual parameters passed for a and b are 5 and 3 respectively, the result is 8. The following line of code:

```
return (r);
```

finalizes function addition, and returns the control back to the function that called it in the first place (in this case, main). At this moment the program follows its regular course from the same point at which it was interrupted by the call to addition. But additionally, because the return statement in function addition specified a value: the content of variable r (return (r));, which at that moment had a value of 8. This value becomes the value of evaluating the function call.

```
int addition (int a, int b)
↓8
z = addition (5 , 3);
```

So being the value returned by a function the value given to the function call itself when it is evaluated, the variable z will be set to the value returned by addition (5, 3), that is 8. To explain it another way, you can imagine that the call to a function (addition (5,3)) is literally replaced by the value it returns (8). The following line of code in main produces the printing of the result on the screen:

```
cout << "The result is " << z;
```

### Scope of variables:

The scope of variables declared within a function or any other inner block is only their own function or their own block and cannot be used outside of them. For example, in the previous example it would have been impossible to use the variables a, b or r directly in function main since they were variables local to function addition. Also, it would have been impossible to use the variable z directly within function addition, since this was a variable local to the function main.

|                                |                 |
|--------------------------------|-----------------|
| #include <iostream.h>          |                 |
| int x=10,y=15;                 | Global Variable |
| int addition (int a, int b)    |                 |
| { int r;                       | Local Variable  |
| r=a+b;                         |                 |
| return (r);                    |                 |
| }                              |                 |
| void main ()                   |                 |
| { int z;                       | Local Variable  |
| z = addition (5,3);            |                 |
| cout << "The result is " << z; |                 |
| z = addition (x,y);            |                 |
| cout << "The result is " << z; |                 |
| }                              |                 |

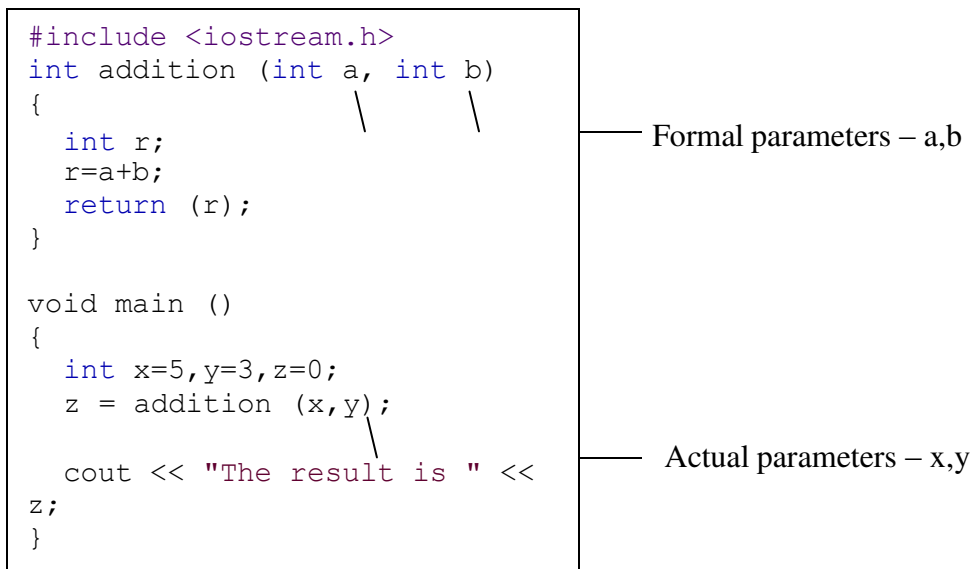
Therefore, the scope of local variables is limited to the same block level in which they are declared. Nevertheless, we also have the possibility to declare global variables; These are visible from any point of the code, inside and outside all functions. In order to declare global variables you simply have to declare the variable outside any function or block; that means, directly in the body of the program.

### Parameters (or Arguments) and their types:

Parameters are the values passed to a function for its working.

There are two types of parameters:

1. **Actual parameters:** Parameters that appears in the calling statement of the function are known as actual parameters.
2. **Formal Parameters:** Parameters that appears in the function prototype or header statement of the function definition are known as formal parameters. **Example :**



### Calling Functions:

There are two ways to call the function by passing the parameters or arguments to a function :

- Call by value / Pass by value
- Call by reference / Pass by reference

#### **Parameters passed by value:**

Until now, in all the functions we have seen, the arguments passed to the functions have been passed *by value*.

When we call a function by passing parameters by value, the values of actual parameters get copied into the formal parameters but not the variables themselves. The changes made in the values of formal parameters will not reflected back to the actual parameters. For example, suppose that we called our first function addition using the following code:

```

int x=5, y=3, z;
z = addition (x , y);

```

What we did in this case was to call to function addition passing the values of x and y, i.e. 5 and 3 respectively, but not the variables x and y themselves.

```

int addition (int a, int b)
 ↑ ↑
z = addition (5 , 3);

```

This way, when the function addition is called, the value of its local variables a and b become 5 and 3 respectively, but any modification to either a or b within the function addition will not have any effect in the values of x and y outside it, because variables x and y were not

themselves passed to the function, but only copies of their values at the moment the function was called.

### Parameters passed by reference:

There might be some cases where we need to manipulate from inside a function the value of an external variable. For that purpose we can use arguments passed by reference, as in the function **duplicate** of the following example:

```
// passing parameters by reference
#include <iostream>
void duplicate (int& a, int& b, int& c)
{ a*=2;
 b*=2;
 c*=2;
}
int main ()
{ int x=1, y=3, z=7;
 duplicate (x, y, z);
 cout << "x=" << x << ", y=" << y << ", z=" << z;
 return 0;
}
```

### Output:

x=2, y=6, z=14

The first thing that should call your attention is that in the declaration of **duplicate** the type of each parameter was followed by an ampersand sign (&). This ampersand is what specifies that their corresponding arguments are to be passed *by reference* instead of *by value*.

When a variable is passed by reference we are not passing a copy of its value, but we are somehow passing the variable itself to the function and any modification that we do to the formal parameters will have an effect in actual parameter passed as arguments in the call to the function.

**void duplicate (int& a,int& b,int& c)**

**duplicate ( x , y , z );**

To explain it in another way, we associate a, b and c with the arguments passed on the function call (x, y and z) and any change that we do on a within the function will affect the value of x outside it. Any change that we do on b will affect y, and the same with c and z.

That is why our program's output, that shows the values stored in x, y and z after the call to **duplicate**, shows the values of all the three variables of main doubled.

If when declaring the following function: `void duplicate (int& a, int& b, int& c)`

we had declared it this way: `void duplicate (int a, int b, int c)`

i.e., without the ampersand signs (&), we would have not passed the variables by reference, but a copy of their values instead, and therefore, the output on screen of our program would have been the values of x, y and z without having been modified. For example,

```
// passing parameters by value
#include <iostream>
void duplicate (int a, int b, int c)
{ a*=2;
 b*=2;
 c*=2;
}
```

```

 cout << "a=" << a << ", b=" << b << ", c=" << c;
}
int main ()
{ int x=1, y=3, z=7;
 duplicate (x, y, z);
 cout << "x=" << x << ", y=" << y << ", z=" << z;
 return 0;
}

```

**Output :**

```

a=2, b=6, c=14
x=1, y=3, z=7

```

**Passing by reference is also an effective way to allow a function to return more than one value.**

**A Comparison:**

| S.N. | Parameters passed by value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Parameters passed by reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.   | When we call a function by passing parameters by value, the values of actual parameters get copied into the formal parameters but not the variables themselves.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | When we call a function by passing parameters by reference, formal parameters create a reference or pointer directly to the actual parameters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 2.   | The changes made in the values of formal parameters will not be reflected back to the actual parameters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | The changes made in the values of formal parameters will be reflected back to the actual parameters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 3.   | <p>Example -</p> <pre> // passing parameters by value #include &lt;iostream.h&gt; void swap (int a, int b) { a=a+b; b=a-   b;   c=a-b;   cout &lt;&lt; "Values inside function \n";   cout&lt;&lt;"a=" &lt;&lt; a &lt;&lt; ", b=" &lt;&lt; b; } int main () { int x=1, y=3;   swap (x, y);   cout &lt;&lt; "Values inside main \n";   cout &lt;&lt; "x=" &lt;&lt; x &lt;&lt; ", y=" &lt;&lt; y;   return 0; } </pre> <p>Output :</p> <pre> Values inside function a=3, b=1 Values inside main x=1, y=3 </pre> <p><b>(Note:</b> Changes made in formal parameters (a and b) are not reflected back to actual parameters (x and y))</p> | <p>Example -</p> <pre> // passing parameters by reference #include &lt;iostream.h&gt; void swap (int &amp;a, int &amp;b) { a=a+b; b=a-   b;   c=a-b;   cout &lt;&lt; "Values inside function \n";   cout&lt;&lt;"a=" &lt;&lt; a &lt;&lt; ", b=" &lt;&lt; b; } int main () { int x=1, y=3;   swap (x, y);   cout &lt;&lt; "Values inside main \n";   cout &lt;&lt; "x=" &lt;&lt; x &lt;&lt; ", y=" &lt;&lt; y;   return 0; } </pre> <p>Output :</p> <pre> Values inside function a=3, b=1 Values inside main x=3, y=1 </pre> <p><b>(Note:</b> Changes made in formal parameters (a and b) are reflected back to actual parameters (x and y))</p> |

**Default values in parameters:**

When declaring a function we can specify a default value for each of the last parameters. This value will be used if the corresponding argument is left blank when calling to the function. To do that, we simply have to use the assignment operator and a value for the arguments in the function declaration. If a value for that parameter is not passed when the function is called, the default value is used, but if a value is specified this default value is ignored and the passed value is used instead. Example:

```
// default values in functions
#include <iostream>
using namespace std;
int divide (int a, int b=2)
{ int r;
 r=a/b;
 return (r);
}
int main ()
{ cout << divide (12);
 cout << endl;
 cout << divide (20,4);
 return 0;
}
```

**Output:**

6  
5

**UNSOLVED PROBLEMS**

**Problem 1:** Find the output of the following program:

```
#include <iostream.h>
void Changethecontent(int Arr[], int Count)
{for (int C = 1;C < Count; C++) Arr[C-
 1] += Arr[C];
}
void main()
{int A[] = {3,4,5}, B[]={10,20,30,40}, C[]={900,1200};
 Changethecontent(A, 3);
 Changethecontent(B, 4);
 Changethecontent(C, 2);
 for (int L = 0;L < 3;L++) cout<<A[L]<< "#";
 cout<<endl;
 for (L = 0;L < 4;L++) cout << B[L] << "#";
 cout << endl;
 for (L = 0;L < 2;L++) cout<<C[L] << "#";
}
```

**Problem 2:** Write a C++ function SUMFUN() having two parameters X (of type double) and n (of type integer) with a result type as double to find the sum of the series given below:

$$X + \frac{X^2}{3!} + \frac{X^3}{5!} + \dots + \frac{X^n}{(2n-1)!}$$

**Problem 3:** Write a function called zero\_Small( ) that has two integer arguments being passed by reference and sets the smaller of the two numbers to 0. Write the main program to access this function.

**Problem 4:** What is the difference between call by value and call by reference? Give an example in C++ to illustrate both.

**Problem 5:** Find the output of the following program:

```
#include<iostream.h>
void Indirect(int Temp=20)
{for (int l=10; I<=Temp; I+=5)
 cout<<I<<" ";
 cout<<endl;
}
void Direct (int &Num)
{ Num+=10;
```



```
 Indirect(Num);
 }
void main()
{
 int Number=20;
 Direct(Number);
 Indirect();
 cout<< " Number=" <<Number<<endl ;
}
```

**LIBRARY FUNCTIONS**

Character functions, String Function, Input/Output Manipulation Functions, Mathematical Functions, Some more header files and their associated functions randomize( ) and random function

**Character Functions:****Header File : ctype.h**

| Function  | Description                                              |
|-----------|----------------------------------------------------------|
| isalnum() | Returns nonzero if a character is alphanumeric           |
| isalpha() | Returns nonzero if a character is alphabetic             |
| isdigit() | Returns nonzero if a character is a digit                |
| islower() | Returns nonzero if a character is lowercase              |
| isupper() | Returns nonzero if a character is an uppercase character |
| tolower() | Converts a character to lowercase                        |
| toupper() | Converts a character to uppercase                        |

**String Functions :****Header File : string.h**

|         |                                                                                                                                                                                                                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| strcat  | concatenates two strings                                                                                                                                                                                              |
|         | compares two strings ( <b>Case insensitive comparison</b> ), and returns one of the following -                                                                                                                       |
| strcmpi | <b>-ve value</b> : if first string comes before second string in dictionary order<br><b>Zero</b> : if both the strings are equal.<br><b>+ve value</b> : if first string comes after second string in dictionary order |
|         | compares two strings ( <b>Case sensitive comparison</b> ), and returns one of the following -                                                                                                                         |
| strcmp  | <b>-ve value</b> : if first string comes before second string in dictionary order<br><b>Zero</b> : if both the strings are equal.<br><b>+ve value</b> : if first string comes after second string in dictionary order |
| strcpy  | copies (overwrite) one string to another                                                                                                                                                                              |
| strlen  | returns the length of a given string                                                                                                                                                                                  |

**Input/Output Manipulation Functions :****Header File : iomanip.h**

|              |                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------|
| setf         | Sets ios flags.                                                                                  |
| setw         | Sets the width of the field assigned for output.                                                 |
| setprecision | Sets the total number of digits to be displayed when floating point numbers are to be displayed. |

**Mathematical Functions :****Header File : math.h**

|       |                                                                                   |
|-------|-----------------------------------------------------------------------------------|
| pow   | Returns base raised to the power passed as first & second argument respectively.. |
| exp   | Returns natural logarithm e raised to the power passed as argument.               |
| fabs  | Returns absolute value of the number passed as argument.                          |
| ceil  | Returns smallest integer not less than the real number passed as argument.        |
| floor | Returns largest integer not greater than the real number passed as argument.      |

fmod Returns remainder of the division  $x/y$ . (  $x, y$  passed as argument)

sqrt Returns square root of the number passed as argument.

### Other functions of math.h:

**Trigonometrical functions:** acos, asin, atan, sinh, cosh, tanh etc

**Exponential functions:** exp, frexp etc

**Logarithmic functions:** log, log10 etc.

### Some more header files and their associated functions

#### stdio.h

|      |         |      |      |         |      |        |        |
|------|---------|------|------|---------|------|--------|--------|
| getc | getchar | gets | putc | putchar | puts | remove | rename |
|------|---------|------|------|---------|------|--------|--------|

#### fstream.h

|       |       |       |         |      |       |     |       |
|-------|-------|-------|---------|------|-------|-----|-------|
| open  | close | get   | getline | read | write | put | seekg |
| seekp | tellg | tellp | eof     |      |       |     |       |

#### stdlib.h

|      |       |        |           |        |
|------|-------|--------|-----------|--------|
| rand | srand | random | randomize | malloc |
|------|-------|--------|-----------|--------|

### randomize()

randomize() initializes the random number generator with a random number.

### Working of random ( ) function

**Syntax :** random (num)

**random ( num)** generates random numbers within the range 0 to (num-1). For example random(6) generates random numbers between 0 to 6.

Example:

In the following program, if the value of N given by the user is 20, what maximum and minimum value the program could possibly display?

```
#include<iostream.h>
#include<stdlib.h>
void main()
{ int N, Guessme;
 randomize();
 cin>>N;
 Guessme = random(N-10) + 10;
 cout<<Guessme<<endl;
}
```

### Solution:

| N  | Guessme                                                                                                                                                                                        |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20 | random(N-10)+10<br>random(20-10)+10<br>random(10)+10<br><b>[Note : here random (10) will generates random numbers between 0 to 10-1 ( 0 to 9)]</b><br>Minimum value : 10<br>Maximum value : 19 |

**Minimum value : 10**

**Maximum value : 19**

## UNSOLVED PROBLEMS

- Problem 1:** Write the names of the header files to which the following belong :  
 puts(), sin(), setw(), sqrt(), strcat(), gets(),  
 strcpy(), abs(), isupper(), pow(), random(), strcmp(),  
 isalnum(), isalpha(), fabs()
- Problem 2:** In the following program, find the correct possible output(s) from the options:
- ```
#include <stdlib.h>
#include <iostream.h>
void main ( )
{
    randomize ( );
    char Area[][10]= {"NORTH", "SOUTH", "EAST", "WEST"};
    int ToGo;
    for(int I = 0; I <3, I++)
    {
        ToGo = random (2) +1;
        Cout<<Areaa[ToGo]<<" : ";
    }
}
```
- Problem 3:** In the following C++ program what is the expected value of Myscore from Options (i) to (iv) given below. Justify your answer.
- ```
#include<stdlib.h>
#include<iostream.h>
void main()
{randomize();
 int Score[] = {25,20,34,56, 72, 63}, Myscore;
 Myscore = Score[2 + random(2)];
 cout<<Myscore<<endl;
}
```
- (i) 25  
 (ii) 34  
 (iii) 20  
 (iv) None of the above
- Problem 4:** In the following program, if the value of N given by the user is 15, what maximum and minimum values the program could possibly display?
- ```
#include <iostream.h>
#include <stdlib.h>
void main()
{
    int N,Guessme;
    randomize();
    cin>>N;
    Guessme=random(N)+10;
    cout<<Guessme<<endl;
}
```
- Problem 5:** Find the output of the following program:
- ```
#include <iostream.h>
#include <ctype.h>
void main()
{ char Text[]= "Mind@Work!";
 for (int I=0; Text[I] != „\0“; I++)
 {if (! isalpha(Text[I]))
 Text[I]="*";
 else if (isupper (Text[I]))
 Text[I]=Text[I]+1;
 else
 Text[i]=Text[I+1];
 }
 cout<<Text;
}
```

## **STRUCTURE**

Structure, using structure in a program, nested structures, array of structure, Passing structure to a function & returning structure from a function.

### **Structure:**

A structure is a group of data elements of different data types grouped together under one name. These data elements, known as *members*. Structures are declared in C++ using the following syntax:

```
struct structure_name
{
 member_type1 member_name1;
 member_type2 member_name2;
 member_type3 member_name3;
 .
 .
} struct_variable_names;
```

where structure\_name is a name for the structure type, struct\_variable\_name can be a set of valid identifiers for objects that have the type of this structure. Within braces { } there is a list with the data members, each one is specified with a type and a valid identifier as its name.

The first thing we have to know is that a data structure creates a new type: Once a data structure is declared, a new type with the identifier specified as structure\_name is created and can be used in the rest of the program as if it was any other type.

Example:-

```
struct product
{
 int weight;
 float price;
} ; //structure data type definition

product apple; //structure object declaration
product banana, melon;
```

We can also declare the structure objects apple, banana and melon at the moment we define the data structure type right at the end of the struct declaration, and before the ending semicolon this way:

```
struct product
{ int weight;
 float price;
} apple, banana, melon;
```

Once we have declared our three objects of a determined structure type (apple, banana and melon) we can operate directly with their members. To do that we use a dot (.) inserted between the object name and the member name. For example, we could operate with any of these elements as if they were standard variables of their respective types:

```
apple.weight
apple.price
banana.weight
banana.price
melon.weight
melon.price
```

## Using structure in a program:

**Example : Program using structure to accept and show details of a student.**

```
#include <iostream.h>
#include<conio.h>
#include<stdio.h>
struct student
{
 int sno;
 char name[20];
 char std[3];
};
void main()
{
 clrscr();
 student s;
 cout<<"\n Enter Student No. : ";
 cin>>s.sno;
 cout<<"\n Enter Student Name : ";
 gets(s.name);
 cout<<"\n Enter Class : ";
 cin>>s.std;
 cout<<"\n Student Details : ";
 cout<<"\n Student No. : "<<s.sno;
 cout<<"\n Student Name : ";puts(s.name);
 cout<<"\n Class : "<<s.std;
 getch();
}
```

## Nested structure:

**Example : Program using nested structure to accept and show details of a student.**

```
#include <iostream.h>
#include<conio.h>
#include<stdio.h>
struct date
{
 int dd;
 int mm;
 int yy;
};
struct student
{
 int sno;
 char name[20];
 char std[3];
 date dob;
};
void main()
{
 clrscr();
 student s;
 cout<<"\n Enter Student No. : ";
 cin>>s.sno;
 cout<<"\n Enter Student Name : ";
 gets(s.name);
 cout<<"\n Enter Class : ";
```

```

cin>>s.std;
cout<<"\n Enter Date of birth : ";
cout<<"\n dd : ";
cin>>s.dob.dd;
cout<<"\n mm : ";
cin>>s.dob.mm;
cout<<"\n yyyy : ";
cin>>s.dob.yy;
cout<<"\n Student Details : ";
cout<<"\n Student No. : "<<s.sno;
cout<<"\n Student Name : ";puts(s.name);
cout<<"\n Class : "<<s.std;
cout<<"\n Date of birth : ";
cout<<"\n dd : "<<s.dob.dd;
cout<<"\n mm : "<<s.dob.mm;
cout<<"\n yyyy : "<<s.dob.yy;
getch();
}

```

### Array of structure:

**Example : Program using array of structure objects to accept and show details of 5 students.**

```

#include <iostream.h>
#include<conio.h>
#include<stdio.h>
struct date
{
 int dd;
 int mm;
 int yy;
};
struct student
{
 int sno;
 char name[20];
 char std[3];
 date dob;
};
void main()
{
 clrscr();
 student s[5];
 for(int i=0;i<=4;i++)
 {
 cout<<"\n Enter Details for Student "<<i+1;
 cout<<"\n Enter Student No. : ";
 cin>>s[i].sno;
 cout<<"\n Enter Student Name : ";
 gets(s[i].name);
 cout<<"\n Enter Class : ";
 cin>>s[i].std;
 cout<<"\n Enter Date of birth : ";
 cout<<"\n dd : ";
 cin>>s[i].dob.dd;
 cout<<"\n mm : ";
 cin>>s[i].dob.mm;
 cout<<"\n yyyy : ";
 }
}

```

```

 cin>>s[i].dob.yy;
}
for(i=0;i<=4;i++)
{
 cout<<"\n Student Details for student "<<i+1;
 cout<<"\n Student No. : "<<s[i].sno;
 cout<<"\n Student Name : ";puts(s[i].name);
 cout<<"\n Class : "<<s[i].std;
 cout<<"\n Date of birth : ";
 cout<<"\n dd : "<<s[i].dob.dd;
 cout<<"\n mm : "<<s[i].dob.mm;
 cout<<"\n yyyy : "<<s[i].dob.yy;
}
getch();
}

```

### Passing structure to a function & returning structure from a function:

#### **Example : Program to find sum of two complex numbers.**

```

#include <iostream.h>
#include<conio.h>
struct complex
{int real;
 int imag;
};
complex sum(complex a, complex b)
{complex c;
 c.real=a.real+b.real;
 c.imag=a.imag+b.imag;
 return c;
}
void main()
{clrscr();
 complex n1,n2,n3;
 n1.real=2;
 n1.imag=3;
 cout<<"\n First Number : "<<n1.real<<" + "<<n1.imag <<"i";
 n2.real=3;
 n2.imag=5;
 cout<<"\n Second Number : "<<n2.real<<" + "<<n2.imag <<"i";
 n3=sum(n1,n2);
 cout<<"\n Sum : "<<n3.real<<" + "<<n3.imag <<"i";
 getch();
}

```

#### **Example: Find the output of the following program -**

```

#include <iostream.>
struct three_d;
{int x,y,z;}
void movein(three_d &t, int step=1)
{t.x + = step;
- = step;
 t.z + = step;
}
void moveout(three_d &t, int step=1)
{t.x - = step;
 t.y + = step;
}

```



```

- = step;
}
void main()
{three_d t1={10,20,5}, t2=(30,10,40);
 movein(t1);
 moveout(t2,5);
 cout<<t1.x<<" "<<t1.y<<" "<<t1.z<<endl;
 cout<<t2.x<<" "<<t2.y<<" "<<t2.z<<endl;
 movein(t2,10);
 cout<<t2.x<<" "<<t2.y<<" "<<t2.z<<endl;
}

```

**Solution:**

```

11, 19, 6
25, 15, 35
35, 5, 45

```

**UNSOLVED PROBLEMS**

**Problem 1:** Find the output of the following program:

```

#include <iostream.h>
struct PLAY
{ int Score, Bonus;
};
void Calculate(PLAY &P, int N=10)
{P.Score++;
 P.Bonus += N;
}
void main()
{PLAY PL={10,15};
 Calculate(PL, 5);
 cout<<PL.Score<<" ":"<<PL.Bonus<<endl;
 Calculate(PL);
 cout<<PL.Score<<" ":"<<PL.Bonus<<endl;
 Calculate(PL,15);
 cout<<PL.Score<<" ":"<<PL.Bonus<<endl;
}

```

**Problem 2:** Give the output of the following program:

```

#include <iostream.h>
struct Pixel
{ int C, R;
};
void Display (Pixel P)
{ cout << "Col" << P.C << "Row" << P.R << endl;
}
void main ()
{ Pixel X={40, 50}, Y, Z;
 Z = X;
 X . C += 10 ;
 Y = Z ;
 Y . C += 10 ;
 Y . R += 20 ;
 Z . C -= 15 ;
 Display (X) ;
 Display (Y) ;
 Display (Z) ;
}

```

## : Object Oriented Programming

### **OOP**

Object Oriented Programming: Introduction, General OOPs concepts, Class, Inheritance, Abstraction, Data Hiding, Encapsulation, Polymorphism

### **Object-oriented programming (OOP):**

Object Oriented Programming is a programming paradigm that uses "objects" - data structures consisting of data fields and methods together with their interactions - to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, modularity, polymorphism, and inheritance.

### **General OOPs Concepts:**

#### **Class:**

A Class is a user defined datatype which contains the variables, properties and methods in it. 'A *class defines*' the abstract characteristics of a thing (object), including its characteristics (its **attributes, fields** or **properties**) and the thing's behaviors (the **things it can do**, or **methods, operations** or **features**).

In other words, A class is a *blueprint* or *factory* that describes the nature of something. For example, the class Dog would consist of traits shared by all dogs, such as breed and fur color (characteristics), and the ability to bark and sit (behaviors).

Classes provide modularity and structure in an object-oriented computer program. A class should typically be recognizable to a non-programmer familiar with the problem domain, meaning that the characteristics of the class should make sense in context. Also, the code for a class should be relatively self-contained (generally using **encapsulation**). Collectively, the properties and methods defined by a class are called **members**.

#### **Inheritance:**

Inheritance is a process in which a class inherits all the state and behavior of another class. This type of relationship is called child-Parent or is-a relationship. "Subclasses" are more specialized versions of a class, which *inherit* attributes and behaviors from their parent classes, and can introduce their own.

Using Inheritance all subclasses of a base class define only their unique properties & methods and reuse the common properties from the base class.

**Note: For details see the topic Inheritance: Extending classes**

#### **Abstraction:**

Abstraction is simplifying complex reality by modeling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem.

#### **Data Hiding:**

Data hiding is a property of OOPS by which the crucial data of an object is made hidden from the rest of the program or outside world. In C++ data hiding is provided by using access specifiers - **Private & Protected**.

**Note : For details see Classes and Objects.**

#### **Encapsulation:**

Encapsulation refers to the wrapping of data and associated functions together under a unit class.

**Note : for examples see Class and Objects**

**Polymorphism:**

Polymorphism in object-oriented programming is the ability of objects belonging to different data types to respond to calls of methods of the same name, each one according to an appropriate type-specific behavior. One method, or an operator such as +, -, or \*, can be abstractly applied in many different situations.

Polymorphism can be implemented using Function / Operator overloading.

**Example:**

```
#include <iostream.h>
int operate (int a, int b)
{ return (a*b);
}
float operate (float a, float b)
{return (a/b);
}
int main ()
{ int x=5,y=2;
 float n=5.0,m=2.0;
 cout << operate (x,y);
 cout << "\n";
 cout << operate (n,m);
 cout << "\n";
 return 0;
}
```

**Output:**

```
10
2.5
```

**UNSOLVED PROBLEMS**

- Problem 1: What do you understand by Polymorphism? Give an example in C++ to show its implementation in C++.
- Problem 2: What do you understand by Data Encapsulation and Data Hiding?
- Problem 3: What is Inheritance? Give an example in C++ to show its implementation in C++.
- Problem 4: Illustrate the concept of Inheritance with the help of an example.
- Problem 5: Encapsulation is one of the major properties of OOP. How is it implemented in C++?
- Problem 6: Reusability of classes is one of the major properties of OOP. How is it implemented in C++?
- Problem 7: Define the term Data Hiding in the context of Object Oriented Programming. Give a suitable example using a C++ code to illustrate the same.

## : Function Overloading

### **FUNCTION OVERLOADING**

Introduction, Examples of Function overloading, Execution of overloaded functions, inline functions

#### **Introduction:**

In C++ two different functions can have the same name if their parameter types or number are different. That means that we can give the same name to more than one function if they have either a different number of parameters or different types in their parameters.

#### **Examples of Function Overloading:**

##### **Example:**

```
//overloaded function
#include <iostream.h>
int operate (int a, int b)
{
 return (a*b);
}
float operate (float a, float b)
{
 return (a/b);
}
int main ()
{ int x=5,y=2;
 float n=5.0,m=2.0;
 cout << operate (x,y);
 cout << "\n";
 cout << operate (n,m);
 cout << "\n";
 return 0;
}
```

##### **Output :**

```
10
2.5
```

##### **Example:**

```
#include<stdio.h>
#include<conio.h>
void print(char *a)
{puts(a);
}
void print(char *a, char *b)
{puts(a);
 puts(b);
}
void print()
{puts("Welcome Guest");
}
void main()
{clrscr();
 char *name1,*name2; // var. declaration
 puts("Enter Your Name : "); //display message
```

```

gets(name1); //accepting string1
puts("Enter Your Father's Name : "); //display message
gets(name2); //accepting string2
print(); //function call
print(name1); //function call
print(name1,name2); //function call
getch();
}

```

### **Execution of overloaded functions:**

The behavior of a call to overloaded function depends on the number and type of the arguments passed. The execution depends on the best match of the number and type of the arguments.

### **Inline functions:**

The inline specifier indicates the compiler that inline substitution is preferred to the usual function call mechanism for a specific function. This does not change the behavior of a function itself, but is used to suggest to the compiler that the code generated by the function body is inserted at each point the function is called, instead of being inserted only once and perform a regular call to it, which generally involves some additional overhead in running time.

The format for its declaration is:

```
inline type name (arguments ...) { instructions ... }
```

and the call is just like the call to any other function. You do not have to include the inline keyword when calling the function, only in its declaration.

Most compilers already optimize code to generate inline functions when it is more convenient. This specifier only indicates the compiler that inline is preferred for this function.

## **UNSOLVED PROBLEMS**

- Problem 1:       What is function overloading?
- Problem 2:       Illustrate the concept of function overloading with the help of an example.
- Problem 3:       What do you understand by function overloading? Give an example illustrating its use in a C++ program.

## : Classes and Objects

### **CLASSES AND OBJECTS**

Class : Introduction and Need, Defining Classes, Access Specifiers, Data members and Member Functions, Creation of objects & accessing members

#### Class:

Collection of objects sharing common properties & behavior.

Or

Simply Collection of similar objects.

#### Need:

To define real world objects more effectively.( i.e. not only data associated with them but also their associated behavior or operations.

#### Defining Classes:

##### **Declaration of Class**

1. Data Members
2. Member Functions
3. Program access levels
4. Class Tagname

##### **Syntax**

```
class < Class Name >
{
 private :
 [Var Declaration]
 [Function Declarations]
 protected :
 [Var Declaration]
 [Function Declarations]
 public :
 [Var Declaration]
 [Function Declarations]
};
```

##### **Example :**

```
class A
{
 int a;
 public:
 void getdata()
 {
 cin>>a;
 }
 void showdata()
 {
 cout<<a;
 }
};
```

**Note:** Functions defined within the scope of the class are by default inline. No need to use inline keyword with them to make them inline.

## Access specifiers:

Access specifiers are used to identify access rights for the data and member functions of the class. There are three main types of access specifiers in C++ programming language:

- **private**
- **public**
- **protected**

### Private:

A *private* member within a class denotes that only members of the same class have accessibility. The *private* member is inaccessible from outside the class.

(Note: Default access specifier, if no access specifier is provided then all the Data members and member functions are treated as private members).

### Public:

*Public* members are accessible from outside the class.

### Protected:

A protected access specifier is a stage between *private* and *public* access. If member functions defined in a class are *protected*, they cannot be accessed from outside the class but can be accessed from the derived class.

While defining access specifiers, the programmer must use the keywords: *private*, *public* or *protected* when needed, followed by a semicolon and then define the data and member functions under it.

## Defining function outside the class (out of scope of class)

Member Functions of a class can be defined outside the class. For this purpose the scope resolution operator (`::`) can be used.

```
Return_type class_name : : function_name(argument list)
{
// statements to be executed
}
```

### Example

```
class A
{
 int a;
 Public:

 void getdata();
 void showdata();
};

void A:: getdata()
{
 Cin>>a;
}

void A:: showdata()
{
 Cout<<a;
}
```

## Use of a Class in a Program

Define the following in a sequence to use a class in a program

1. Class Definition
2. Class Method
3. In main() Create Object

### Example:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class stud
{private: //private members
 int sno;
 char sname[80];
 int cls;
public: //public members
 void input(void);
 void output(void);
};
void stud :: input(void)
{
 cout<<"Enter Sno : ";
 cin>>sno;
 cout<<"Enter Sname : ";
 gets(sname);
 cout<<"\nEnter Class :";
 cin>>cls;
}
void stud :: output(void)
{
 cout<<"\n Sno : "<<sno;
 cout<<"\n Sname : "<<sname;
 cout<<"\n Class : "<<cls;
}
void main()
{
 clrscr();
 stud s;
 s.input();
 s.output();
 getch();
}
```

### Example :

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class stud
{
 int sno;
 char sname[80];
public:
 void input(void);
 void output(void);
 int cls;
```



```

};
void stud :: input(void)
{
 cout<<"Enter Sno : ";
 cin>>sno;
 cout<<"Enter Sname : ";
 gets(sname);
}
void stud :: output(void)
{
 cout<<"\n Sno : "<<sno;
 cout<<"\n Sname : "<<sname;
 cout<<"\n Class : "<<cls;
}
void main()
{
 clrscr();
 stud s;
 s.input();
 cout<<"\nEnter Class :";
 cin>>s.cls;/**cls - Being a public member available outside the
class
 cout<<"\nEnter S.No :";
 cin>>s.sno;/**Program gives error as sno - Being a private member
not available outside the class */
 s.output();
 getch();
}

```

### Important Points:

- The private members provide data hiding by preventing us from accessing the data directly
- The class declaration must end with a semi-colon.
- It is important to include the corresponding *.h* file; otherwise, we will get compile-time errors. In the *#include*, the name of the file is enclosed in quotes, not in angle brackets. Angle brackets are used for including standard library header files, and quotes are used for including our own header files (this tells the compiler where to look for the header file).

#### Note

A function declared using the friend keyword is not considered a member of the class in which it is declared a friend (although it can be a member of another class). A friend declaration controls the access a nonmember function has to class data.

### Creation of Objects:

Once the class is created, one or more objects can be created from the class as objects are instance of the class. Just as we declare a variable of data type *int* as:

```
int x;
```

Objects are also declared as:

```
class name followed by object name;
```

```
ex e1;
```

This declares *e1* to be an object of class *ex*.

For example a complete class and object declaration is given below:

```

class ex
{
 private:
 int x,y;
 public:

```

```

 void sum()
 {

 }
};
main()
{
 ex e1;

}

```

The object can also be declared immediately after the class definition. In other words the object name can also be placed immediately before the closing flower brace symbol } of the class declaration.

### Example

```

class ex
{
 private:
 int x,y;
 public:
 void sum()
 {

 }
}e1 ;

```

## UNSOLVED PROBLEMS

- Problem 1:** Define a class TEST in C++ with following description:  
**Private Members**  
 TestCode of type integer  
 Description of type string  
 NoCandidate of type integer  
 CenterReqd (number of centers required) of type integer  
 A member function CALCNTR() to calculate and return the number of centers as (NoCandidates/100+1)  
**Public Members**  
 A function SCHEDULE() to allow user to enter values for TestCode, Description, NoCandidate & call function CALCNTR() to calculate the number of Centres  
 A function DISPTTEST() to allow user to view the content of all the data members
- Problem 2:** Define a class Tour in C++ with the description given below :  
**Private Members :**  
 TCode of type string  
 NoofAdults of type integer  
 NoofKids of type integer  
 Kilometres of type integer  
 TotalFare of type float  
**Public Members :**  
 A constructor to assign initial values as follows :  
     TCode with the word "NULL"  
     NoofAdults as 0  
     NoofKids as 0  
     Kilometres as 0  
     TotalFare as 0  
 A function AssignFare ( ) which calculates and assigns the value of the data

member TotalFare as follows

For **each** Adult

| Fare(Rs) | For Kilometres |
|----------|----------------|
| 500      | >=1000         |
| 300      | <1000 &>=500   |
| 200      | <500           |

For **each** Kid the above Fare will be 50% of the Fare mentioned in the above table

For example :

If Kilometres is 850, NoofAdults = 2 and NoofKids = 3

Then TotalFare should be calculated as

NumofAdults \* 300 + NoofKids \* 150

i.e.  $2*300 + 3*150=1050$

A function EnterTour( ) to input the values of the data members TCode, NoofAdults, NoofKids and Kilometres; and invoke the Assign Fare( ) function.

A function ShowTour( ) which displays the content of all the data members for a Tour.

Problem 3:

Define a class named HOUSING in C++ with the following descriptions: Private Members:

|        |                              |
|--------|------------------------------|
| REG_NO | integer (Ranges 10- 1000)    |
| NAME   | Array of characters (String) |
| TYPE   | Character                    |
| COST   | Float                        |

Public Members:

Function Read\_Data() to read an object of HOUSING type.

Function Display() to display the details of an object.

Function Draw\_Nos() to choose and display the details of 2 houses selected randomly from an array of 10 objects of type HOUSING. Use random function to generate the registration nos. to match with REG\_NO from the array.

## : Constructors And Destructors

### **CONSTRUCTORS AND DESTRUCTORS**

Constructors, Characteristics of constructors, Types of constructors, Destructors

#### **Constructors:**

Constructors are member functions of a class which are used to initialize the data members of the class objects. These functions are automatically called when an object of its class is created. There is no need to call these functions.

#### **Characteristics of constructors:**

- The name of a constructor is same as that of class in which it is declared.
- Constructors do not have any return type, not even void.
- Constructors are always defined in the public section of the class.
- They cannot be inherited, though a derived class can call the base class constructor.
- A constructor may not be static.
- Like other C++ functions, constructors can also have default arguments.
- It is not possible to take the address of a constructor.
- Member functions may be called
- Constructors are not called directly
- Constructors show polymorphism in a class

#### **Types of Constructors:**

There are three types of constructors.

1. Default Constructors
2. Parameterized Constructors
3. Copy Constructors

##### **1. Default Constructors**

A constructor that accepts no parameter is called the default constructors.

Example

```
class stud
{
 int sno;
 char sname[40];
public :
 stud() // Default Constructor
 {
 sno=0;
 strcpy(sname, "new");
 }
 void getinfo()
 { cin>> sno;
 gets(sname);
 }
 void showinfo()
 { cout<< sno;
 puts(sname);
 }
};
```

The above class stud has sno and sname as data members and three member functions i.e. stud (), getinfo(), showinfo()

Here stud ( ) is a default constructor since having the same name as that of class and does not accept any argument, also declared in the public section.

As we declare the object of this class it will immediately call to the constructor of the class. It automatically assigns the value 0 to variable sno and a "new" to sname.

Here consider the following main function

```
void main()
{
 stud obj; // Default constructor called
 Obj.showinfo(); // displays the value of sno as 0 and sname as
 "new"
 Obj.getinfo(); // reads the user given value from the user
 Obj.showinfo(); // displays the user given values
}
```

The default constructors are very useful when we want to create objects without having to type the initial values.

With a default constructor objects are created just the same way as variables of other data types are created.

## 2. Parameterized Constructors

A constructor that accepts parameters for its invocation is known as parameterized constructors.

Example

```
class stud
{
 int sno;
 char sname[40];
public :
 stud(int s, char n[]) // Parameterized Constructor
 {
 sno=s;
 strcpy(sname,n);
 }
 void getinfo()
 { cin>> sno;
 gets(sname);
 }
 void showinfo()
 { cout<< sno;
 puts(sname);
 }
};
```

This means we always specify the arguments whenever we declare an instance ( object) of the class.

```
void main()
{
 stud obj (1, "Ashu"); // Parameterized constructor invoked
 Obj.showinfo(); // displays the value of sno as 1 and sname as
 "Ashu"
 Obj.getinfo(); // reads the user given value from the user
 Obj.showinfo(); // displays the user given values
}
```

Just like any other function a parameterized constructor can also have default arguments

```
stud(int s=0, char n[]="new\0")
```

- A constructor with default arguments is equivalent to a default constructor.
- A class must not have a default arguments constructor and default constructor together as it generates ambiguity.

### 3. Copy Constructors:

A copy constructor is a constructor of the form `classname( & classname)`. It is used to initialize an object with the values of another object.

The compiler will use the copy constructor whenever -

- We initialize an instance using values of another instance of same type.
- A function returns an object
- A function receives an object as parameter.

```
class stud
{
 int sno;
 char sname[40];
Public :
 stud(stud &s) // Copy Constructor
 {
 sno=s.sno;
 strcpy(sname,s.name);
 }
 stud() // Default Constructor
 {
 sno=0;
 strcpy(sname,"new");
 }
 void getinfo()
 { cin>> sno;
 gets(sname);
 }
 void showinfo()
 { cout<< sno;
 puts(sname);
 }
};

void main()
{
 stud obj; // Default constructor called
 Obj.showinfo(); // displays the value of sno as 0 and sname as "new"
 stud objnew(obj); // Copy constructor invoked and initialize the members
 // of object objnew with the // values of object obj
 // i.e. 0 and "new".
 Objnew.showinfo(); //displays the value of sno as 0 and sname as "new"
 Obj.getinfo(); //reads the user given value for object obj
 Obj.showinfo(); // displays the user given values for object obj
 Objnew.getinfo(); // reads the user given value for object objnew
 Objnew.showinfo(); // displays the user given values for object objnew
}
```

Lets have a look:

```
stud obj; // default constructor used

stud obj1(23, "Nisha"); // parameterized constructor used

stud obj2 = obj; // copy constructor used

stud obj3= obj1; // copy constructor used
```

**Points to remember:**

- Declaring a constructor with arguments hides the default constructor.
- A Constructor with default arguments is equivalent to a default constructor.

- A constructor declared under private access specifier, makes the class private and object of a private class cannot be created.
- A class must not have a default arguments constructor and default constructor together as it generates ambiguity.
- Constructors also show the polymorphism as a single class can have multiple constructors of different forms. (Also known as constructor / function overloading.)

### **Destructors:**

A destructor is a class member function that has the same name as the constructor (and the class ) but with a ~ (tilde) in front.

```
~stud();
```

Destructor is used to deinitialize or destroy the class objects. When an object goes out of scope, its destructor is automatically invoked to destroy the object.

Example :

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
class stud
{
 int sno;
 char sname[40];
public :
 stud() // Default Constructor
 {
 sno=0;
 strcpy(sname,"new");
 cout<<"\nConstructing the object.....";
 }
 ~stud() // Destructor
 {
 cout<< "\nDestructing the object.....";
 }
}
void getinfo()
{cout<<"\nEnter Student No. :";
 cin>> sno;
 cout<<"\nEnter Student Name :";
 gets(sname);
}
void showinfo()
{cout<< "\n Student No.: "<<sno;
 cout<<"\n Student Name :";
 puts(sname);
}
};
void main()
{clrscr();
 stud obj; // Default constructor called
 obj.showinfo(); // displays the value of sno as 0 and sname as
"new"
 obj.getinfo(); // reads the user given value for object obj
 obj.showinfo(); // displays the user given values for object
obj
 getch();
}
```

**Output:**

The screenshot shows the Turbo C++ IDE's Output window. The text displayed is as follows:

```

Constructing the object.....
Student No.: 0
Student Name :new
Enter Student No. :1
Enter Student Name :Jayati
Student No.: 1
Student Name :Jayati
Destructing the object.....

```

Note: To see the effect of destructor open the output window again after the complete execution of program.

### UNSOLVED PROBLEMS

Problem 1: Answer the questions (i) and (ii) after going through the following program:

```

class Match
{int Time;
public:
Match() //Function 1
{Time=0;
cout<<"Match commences"<<endl;
}
void Details() //Function 2
{
cout<<"Inter Section Basketball Match"<<endl;
}
Match(int Duration) //Function 3
{
Time=Duration;
cout<<"Another Match begins now"<<endl;
}
Match(Match &M) //Function 4
{
Time=M.Duration;
cout<<"Like Previous Match " <<endl;
}};

```

- (i) Which category of constructor - **Function 4** belongs to and what is the purpose of using it?
- (ii) Write statements that would call the member Functions 1 and 3.

Problem 2: What is the use of a constructor function in a class? Give a suitable example of a constructor function in a class.

Problem 3: What do you understand by default constructor and copy constructor functions used in classes? How are these functions different from normal constructors?

Problem 4: Differentiate between default constructor and copy constructor, give suitable examples of each.



Problem 5: What is copy constructor? Give an example in C++ to illustrate copy constructor.

Problem 6: Answer the questions (i) and (ii) after going through the following program:

```
#include <iostream.h>
#include<string.h>
class Bazar
{
 char Type[20];
 char Product[20];
 int Qty;
 float Price;
 Bazar() //Function 1
 {
 strcpy (Type, "Electronic");
 strcpy(Product, "Calculator");
 Qty = 10;
 Price = 225;
 }
public :
 void Disp() //Function 2
 {cout << Type << "-<< Product << ".*" << Qty
 << "@<< Price << endl;
 }
};
void main()
{ Bazar B ; //Statement 1
 B. Disp(); //Statement 2
}
```

- Will Statement 1 initialize all the data members for object B with the values given in the Function 1 ? (Yes OR No). Justify your answer suggesting the correction(s) to be made in the above code.(**Hint:** Based on the characteristics of Constructor declaration)
- What shall be the possible output when the program gets executed? (Assuming, if required the suggested correction(s) are made in the program).

Problem 7: Answer the questions (a) and (b) after going through the following class:

```
class Interview
{
 Int Month:
 Public:
 Interview(int y) { Month=y;}
 Interview(Interview & t);
};
```

reate an object, such that it invokes Constructor 1.

(b) Write complete definition for Constructor 2.

Problem 8: Answer the questions (i) and (ii) after going through the following program:

```
class Match
{int Time;
 public:
 Match() //Function 1
 {Time=0;
 cout<<"Match commences"<<endl;
 }
 void Details() //Function 2
 {cout<<"Inter Section Basketball Match"<<endl;
 }
 Match(int Duration) //Function 3
 {Time=Duration;
 cout<<"Another Match begins now"<<endl;
 }
 Match(Match &M) //Function 4
```

```

 {Time=M.Duration;
 cout<<"Like Previous Match "<<endl;
 }
};

```

(i) Which category of constructor - **Function 4** belongs to and what is the purpose of using it?

(ii) Write statements that would call the member Functions 1 and 3.

Problem 9: What is a copy constructor? What do you understand by constructor overloading?

Problem 10: What is default constructor? How does it differ from destructor?

Problem 11: Why is a destructor function required in classes? Illustrate with the help of an example.

Problem 12: Answer the questions (i) and (ii) after going through the following class:

```

class Science
{char Topic[20];
 int Weightage;
public:
 Science () //Function 1
 {
 strcpy (Topic, "Optics");
 Weightage = 30;
 cout<<"Topic Activated";
 }
 ~Science() //Function 2
 {
 cout<<"Topic Deactivated";
 }
};

```

(i) Name the specific features of class shown by Function 1 and Function 2 in the above example.

(ii) How would Function 1 and Function 2 get executed?

## : Inheritance: Extending Classes

### **INHERITANCES**

Inheritance, Need for inheritance, Types of Inheritance, Visibility Mode (Visibility/Accessibility of Inherited Base class Member in Derived class), Implementing Inheritance, Accessibility of members by an object of a derived class, Object's byte size.

### **Inheritance:**

Creating or deriving a new class using another class as a base is called inheritance in C++. The new class created is called a **Derived class** and the old class used as a base is called a **Base class** in C++ inheritance terminology.

The derived class will inherit all the features of the base class in C++ inheritance. The derived class can also add its own features, data etc., It can also override some of the features (functions) of the base class, if the function is declared as **virtual** in base class.

### **Need for Inheritance:**

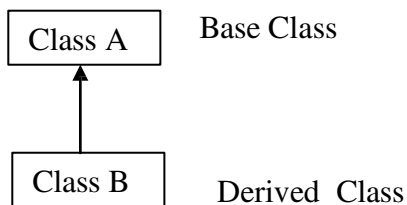
Inheritance is one of the important concepts of object-oriented language. There are several reasons why this concept was introduced in object oriented language. Some major reasons are:

- (i) The capability to express the inheritance relationship which ensures the closeness with the real world model.
- (ii) Idea of reusability, i.e., the new class can use some of the features of old class.
- (iii) Transitive nature of inheritance, i.e., it can be passed on further

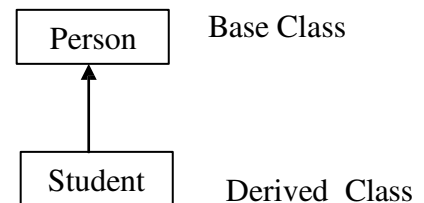
### **Types of Inheritance:**

#### **1. Single Inheritance:**

A derived class inherits properties from a single base class.

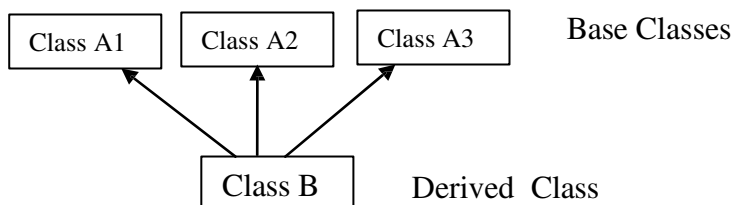


Ex-

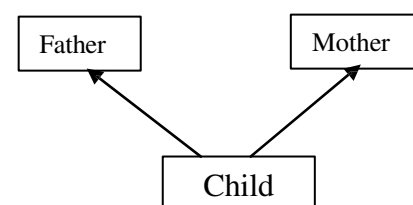


#### **2. Multiple Inheritance:**

A derived class inherits properties from two or more base classes.

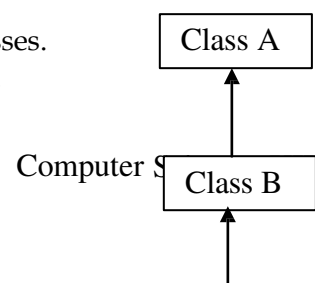


Ex -



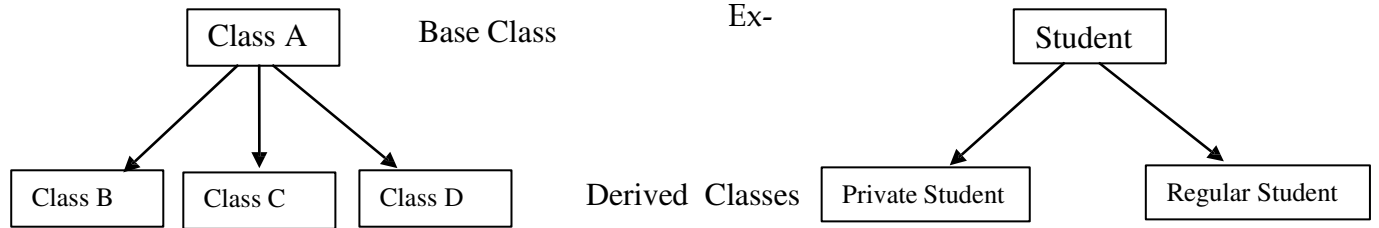
#### **3. Multilevel Inheritance:**

In this type of inheritance, A derived class acts as a base class for other classes. For example, class A inherited in class B and class B is inherited in class C.



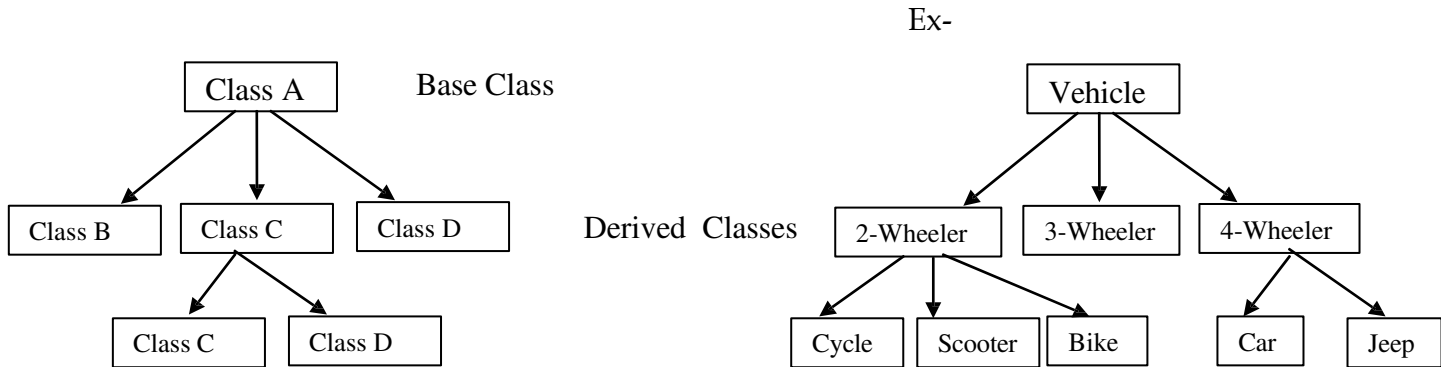
#### 4. Hierarchical Inheritance:

In this type of inheritance a base class has a number of derived classes.



#### 5. Hybrid Inheritance:

In this type of inheritance, we can have mixture of number of inheritances.



**(Combination of Multilevel & Hierarchical Inheritance)**

#### Visibility Mode:

- Private - Public and Protected members of base class inherited as the Private members of the derived class.
- Protected - Public and Protected members of base class inherited as the Protected members of the derived class.
- Public - Protected members of base class inherited as the Protected members of the derived class and Public members of base class inherited as the Public members of the derived class.

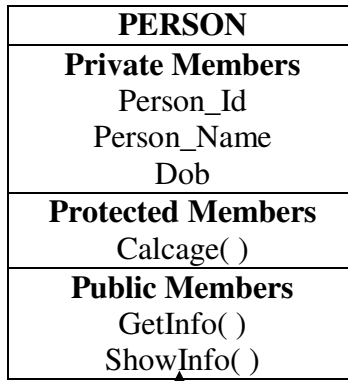
#### Visibility/Accessibility of Inherited Base class Member in Derived class:

Consider the following chart to find which members are accessed from the derived class object after inheritance

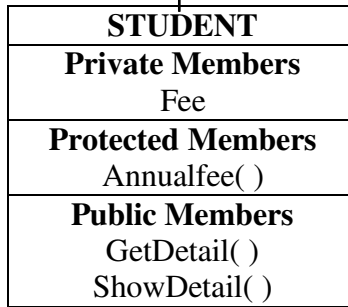
|                       |           | Inheritance Mode         |           |         |
|-----------------------|-----------|--------------------------|-----------|---------|
|                       |           | public                   | protected | private |
| Members in Base Class | public    | public                   | protected | private |
|                       | protected | protected                | protected | private |
|                       | private   | X                        | X         | X       |
|                       |           | Members in derived class |           |         |

**Example**

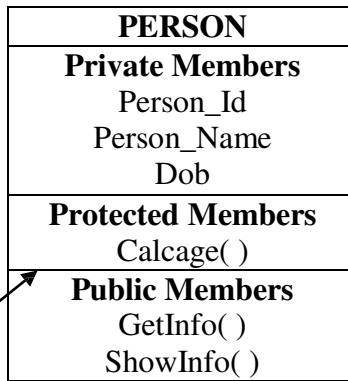
**Base Class**



**Derived**



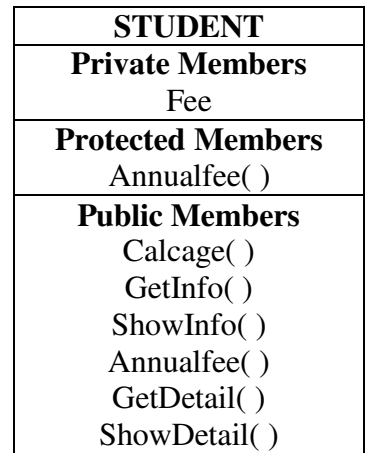
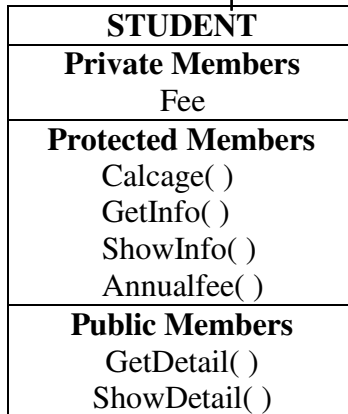
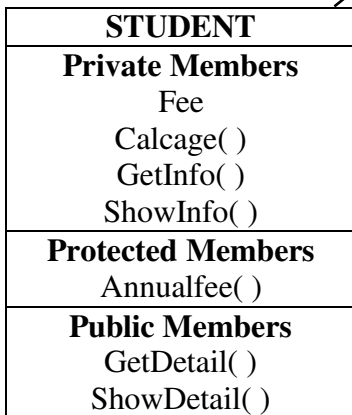
**Class**



**PRIVATE**

**PROTECTED**

**PUBLIC**



## Implementing Inheritance:

### Syntax:

#### Single Inheritance

```
class < Derived Class Tag Name > : <Visibility Mode> <Base Class>
{
 Derived Class Definition
};
```

#### Multiple Inheritance

```
class < Derived Class Tag Name > : <Vis. Mode> <Base Class-1>, <Vis.
Mode> <Base Class-2>,, <Vis. Mode> <Base Class-n>
{
 Derived Class Definition
};
```

#### Example of Single Inheritance:

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
class person
{
 int id;
 char name[20];
 public :
 void getinfo()
 {
 cout<<"\n enter id : ";cin>>id;
 cout<<"\n enter name : "; gets(name);
 }
 void showinfo()
 {
 cout<<"\nid : "<<id;
 cout<<"\nname : "; puts(name);
 }
};
class stud : private person
{
 float fee;
 public :
 void getdetail()
 {getinfo();
 cout<<"\nenter monthly fee : ";cin>>fee;
 }
 void showfee()
 {
 showdetail();
 cout<<"\nMonthly fee :"<<fee;
 cout<<"\nAnnual Fee :"<<fee*12;
 }
};
void main()
{
 clrscr();
 stud s;
 s.getdetail();
 s.showdetail();
 getch();
}
```

**Example of Multiple Inheritance:**

```

#include<iostream.h>
#include<stdio.h>
#include<conio.h>
class person
{
 int id;
 char name[20];
public :
 int cls;
 void getinfo()
 {
 cout<<"\n enter id : ";cin>>id;
 cout<<"\n enter name : "; gets(name);
 }
 void showinfo()
 {
 cout<<"\nid : "<<id;
 cout<<"\nname : "; puts(name);
 }
};
class stud
{
 char course[3];
public:
 void getcourse()
 {
 cout<<"\nEnter course id : "; gets(course);
 }
 void showcourse()
 {
 cout<<"\n Course : ";puts(course);
 }
};
class partstud : private person,private stud
{
 float fee;
public :
 void getfee()
 {
 getinfo();
 getcourse();
 cout<<"\nenter monthly fee : ";cin>>fee;
 }
 void showfee()
 {
 showinfo();
 showcourse();
 cout<<"\nMonthly fee :"<<fee;
 cout<<"\nAnnual Fee :"<<fee*12;
 }
};
void main()
{
 clrscr();
 partstud s;
 s.getfee();
 s.showfee();
 getch();
}

```

**Example of Multilevel Inheritance:**

```

#include<iostream.h>
#include<stdio.h>
#include<conio.h>
class person
{
 int id;
 char name[20];
 public :
 int cls;
 void getinfo()
 {
 cout<<"\n enter id : ";cin>>id;
 cout<<"\n enter name : "; gets(name);
 }
 void showinfo()
 {
 cout<<"\nid : "<<id;
 cout<<"\nname : "; puts(name);
 }
};
class stud : public person
{
 char course[3];
 public:
 void getcourse()
 {
 getinfo();
 cout<<"\nEnter course id : "; gets(course);
 }
 void showcourse()
 {
 showinfo();
 cout<<"\n Course : ";puts(course);
 }
}
class partstud : private stud /*Here class partstud will
automatically inherit all the properties of class person due to the
transitive nature of Inheritance */
{
 float fee;
 public :
 void getfee()
 {getcourse();
 cout<<"\nenter monthly fee : ";cin>>fee;
 }
 void showfee()
 {
 showcourse();
 cout<<"\nMonthly fee :"<<fee;
 cout<<"\nAnnual Fee :"<<fee*12;
 }
};
void main()
{
 clrscr();
 partstud s;
 s.getfee();
 s.showfee();
 getch();
}

```



### Accessibility of members by an object of a derived class:

After derivation all the data members and member functions available under public mode of derived class are directly accessible by object of a derived class.

#### **In the above example -**

##### **Object of stud class can access -**

Data Member - cls

Member Functions - getinfo(),showinfo(),getcourse(),showcourse()

##### **Object of partstud class can access -**

Data Member - None

Member Functions - getfee(),showfee()

#### **Object's byte size**

**Total bytes occupied by an object of a derived class is the sum of the bytes occupied by all the data members of the Base Class(es) and Derived Class.**

Practically it can be **observed by using sizeof() operator**. See the Example C++ code given below -

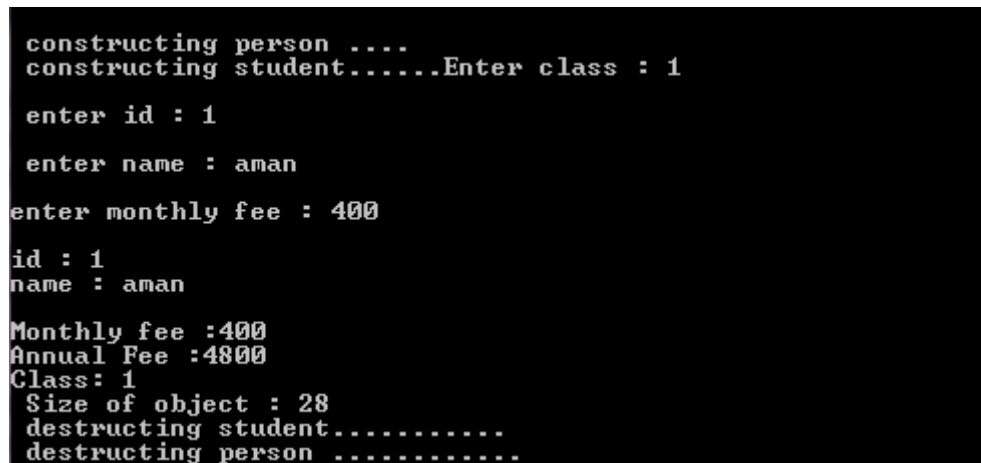
Here total Bytes occupied by the object of stud = sizeof(id) +sizeof(name)+sizeof(cls)+sizeof(fee)  
 =2 bytes+20 bytes+2 bytes+4 bytes  
 =28 bytes

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
class person
{
 int id;
 char name[20];
public :
 person(int k,int m)
 {
 cout<<"\n constructing person";
 id=k;
 cls=m;
 }
 ~person()
 {
 cout<<"\n destructing person";
 }
 int cls;
 void getinfo()
 {
 cout<<"\n enter id : ";cin>>id;
 cout<<"\n enter name : "; gets(name);
 }
 void showinfo()
 {
 cout<<"\nid : "<<id;
 cout<<"\nname : "; puts(name);
 }
};
class stud : private person
{
 float fee;
public :
 person::cls;
 stud(int x,int z, float p) : person(x,z)
```

```

 {
 fee=p;
 cout<<"\n constructing student.....";
 }
 ~stud()
 {
 cout<<"\n destructing student.....";
 }
 void getfee()
 {getinfo();
 cout<<"\nenter monthly fee : ";cin>>fee;
 }
 void showfee()
 {
 showinfo();
 cout<<"\nMonthly fee :"<<fee;
 cout<<"\nAnnual Fee :"<<fee*12;
 }
};
void main()
{
clrscr();
stud s(1,2,25.00);
cout<<"Enter class : ";cin>>s.cls;
s.getfee();
s.showfee();
cout<<"\nClass: "<<s.cls;
cout<<"\n Size of object : "<<sizeof(s);
getch();
}

```



```

constructing person
constructing student.....Enter class : 1

enter id : 1

enter name : aman

enter monthly fee : 400

id : 1
name : aman

Monthly fee :400
Annual Fee :4800
Class: 1
Size of object : 28
destructing student.....
destructing person

```

Some of the exceptions to be noted in C++ inheritance are as follows.

- The constructor and destructor of a base class are not inherited.
- At the time of creation of object of a derived class, first base class constructor is executed and then derived class constructor is executed.

At the time of destruction of object first Derived class destructor is executed and then Base class destructor is executed.

```

 {
 fee=p;
 cout<<"\n constructing student.....";
 }
 ~stud()
 {
 cout<<"\n destructing student.....";
 }
void getfee()
 {getinfo();
 cout<<"\nenter monthly fee : ";cin>>fee;
 }
void showfee()
 {
 showinfo();
 cout<<"\nMonthly fee :"<<fee;
 cout<<"\nAnnual Fee :"<<fee*12;
 }
};
void main()
 {
 clrscr();
 stud s(1,2,25.00);
 cout<<"Enter class : ";cin>>s.cls;
 s.getfee();
 s.showfee();
 cout<<"\nClass: "<<s.cls;
 cout<<"\n Size of object : "<<sizeof(s);
 getch();
 }

```

```

constructing person
constructing student.....Enter class : 1
enter id : 1
enter name : aman
enter monthly fee : 400
id : 1
name : aman
Monthly fee :400
Annual Fee :4800
Class: 1
Size of object : 28
destructing student.....
destructing person

```

Some of the exceptions to be noted in C++ inheritance are as follows.

- The constructor and destructor of a base class are not inherited.
- At the time of creation of object of a derived class, first base class constructor is executed and then derived class constructor is executed.

At the time of destruction of object first Derived class destructor is executed and then Base class destructor is executed.

## UNSOLVED PROBLEMS

- Problem 1: Differentiate between Protected and Private members of a class in context of Inheritance using C++.
- Problem 2: Define Multilevel and Multiple inheritances in context of Object Oriented Programming. Give suitable example to illustrate the same.
- Problem 3: Answer the questions (i) to (iv) based on the following code :
- ```

class Teacher
{ char TNo[5], TName[20], DeptfIO];
  int Workload;
protected:
  float Salary;
  void AssignSal(float);
public:
  Teacher() ;
  void TEntry() ;
  void TDisplay( );
};
class Student
{ char Admno[10], SName[20], Stream[10];
protected:
  int Attendance, TotMarks;
public:
  Student( );
  void SEntry( );
  void SDisplay( );
};
class School : public Student, public Teacher
{ char SCode[10], SchName[20];
public:
  School ( ) ;
  void SchEntry( );
  void SchDisplay( );
};

```
- (i) Which type of Inheritance is depicted by the above example?
- (ii) Identify the member function(s) that cannot be called directly from the objects of class School from the following TEntry()
SDisplay()
SchEntry()
- (iii) Write name of all the member(s) accessible from member functions of class School.
- (iv) If class School was derived privately from class Teacher and privately from class Student, then, name the member function(s) that could be accessed through Objects of class School.
- Problem 4: Differentiate between private and protected visibility modes in context of Object Oriented Programming using a suitable example illustrating each.
- Problem 5: What do you understand by visibility modes in class derivations? What are these modes?
- Problem 6: Answer the questions (i) to (iv) based on the following:
- ```

class CUSTOMER
{
 int Cust_no;
 char Cust_Name[20];
protected:
 void Register();
public:
 CUSTOMER();
 void Status();
};
class SALESMAN
{
 int Salesman_no;
 char Salesman_Name[20];
protected:
 float Salary;
};

```

```

public:
 SALESMAN();
 void Enter();
 void Show();
};
class SHOP : private CUSTOMER , public SALESMAN
{
 char Voucher_No[10];
 char Sales_Date[8];
public:
 SHOP();
 void Sales_Entry();
 void Sales_Detail();
};

```

(i) Write the names of data members which are accessible from objects belonging to class CUSTOMER.

(ii) Write the names of all the member functions which are accessible from objects belonging to class SALESMAN.

Write the names of all the members which are accessible from member functions of class SHOP.

(iv) How many bytes will be required by an object belonging to class SHOP?

Problem 7:

Answer the questions (a) to (d) based on the following:

```

class PUBLISHER
{
 class Pub[12];
 double Turnover;
protected:
 void Register();
public:
 PUBLISHER();
 void Enter();
 void Display();
};
class BRANCH
{
 char CITY[20];
protected:
 float Employees;
public:
 BRANCH();
 void Haveit();
 void Giveit();
};
class AUTHOR: private BRANCH, public PUBLISHER
{
 int Acode;
 char Aname[20];
 float Amount;
public:
 AUTHOR();
 void Start();
 void Show();
};

```

(i) Write the names of data members, which are accessible from objects belonging to class AUTHOR.

(ii) Write the names of all the member functions which are accessible from objects belonging to class BRANCH.

(iii) Write the names of all the members which are accessible from member functions of class AUTHOR.

(iv) How many bytes will be required by an object belonging to class AUTHOR?

## : Data File Handling

### **DATA FILE HANDLING**

Introduction, Using stream I/O classes, Reading from and writing to files using the I/O classes, Using file I/O, Operations on files, some other important functions, file MODES, Differences and definition

#### **Introduction:**

First of all we shall describe the inheritance hierarchy of the stream I/O classes in the C++ standard library by introducing the inheritance relationships amongst the various classes that we will use later in the course when we discuss inheritance. You will not need to understand the details of inheritance to do the lab. For those of you doing Assignment 1 (Weather data) you may be familiar with much of this already, but keep in mind that for students choosing to do only core material this will be new material and the labs are core.

#### **Using stream I/O classes:**

In previous labs when we performed input or output of information in our programs, we used `cin` and `cout`. These are built-in variables (part of the namespace `std`) whose types are designed to support input and output, respectively. When we did this, `cin` allowed us to read data from an input stream that connects the keyboard to our programs.

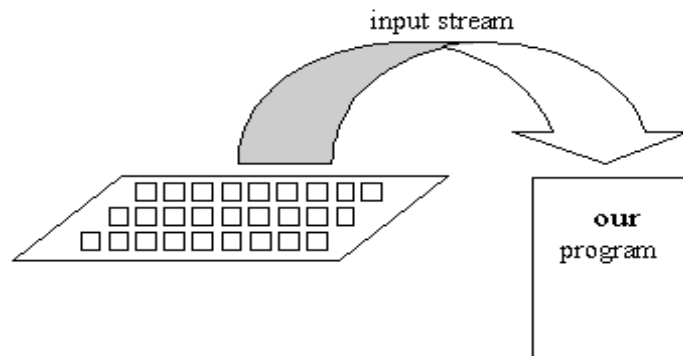


Figure 1: Using the `cin` object for the input stream.

In a similar manner, `cout` allowed us to write information to an output stream that connects our program to the monitor screen.

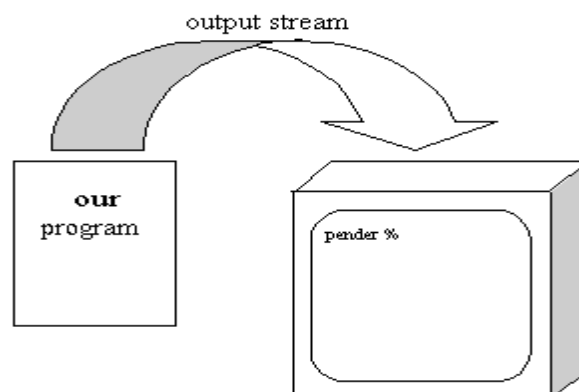


Figure 2: Using the `cout` object for the output stream.

C++ implements these input and output pipes using the stream classes `istream` and `ostream`, respectively. This means that `cin` is an object of class type `istream` and `cout` is an object of

class type `ostream`. These two objects are somewhat special because they have already been declared for us in the file called `iostream` and in the `std` namespace. As long as we include this header file in our programs and state that we are using the `std` namespace, we can make use of `cin` and `cout` in our program.

To do so, we include the following statements in our programs:

```
#include <iostream>
```

```
using namespace std;
```

The classes `istream` and `ostream` are both subclasses that are derived from another base class called `ios`, whose member functions allow basic operations to be performed on input and output streams. These basic operations include obtaining the state of a stream, checking whether an error has occurred while manipulating the stream, and a number of other operations. The derived classes `istream` and `ostream` add stream-input and stream-output operations, respectively, to the set of basic operations provided in the base class `ios`. (For each derived class you can think of the methods that are available as being the union of the derived classes methods with those of the base class, with some caveats for methods that have the same name/signature.)

The terms "base class" and "derived class" are part of the terminology used to describe inheritance relationships in object-oriented languages. We will not be concerned with the details of inheritance until later in the course. For this lab we are simply interested in how to use the two classes `istream` and `ostream`.

Examples of stream-input operations that an object of type `istream` can perform are `get()`, `getline()`, and the input extraction operator `>>`.

Examples of stream-output operations that an object of type `ostream` can perform are `put()`, `write()`, and the output insertion operator `<<`.

The class `iostream` is derived by multiple inheritance from both the `istream` and the `ostream` classes.

The inheritance hierarchy we have just described is illustrated by the following diagram. As you might guess, the class `iostream` allows both reading (input) and writing (output) for a file.

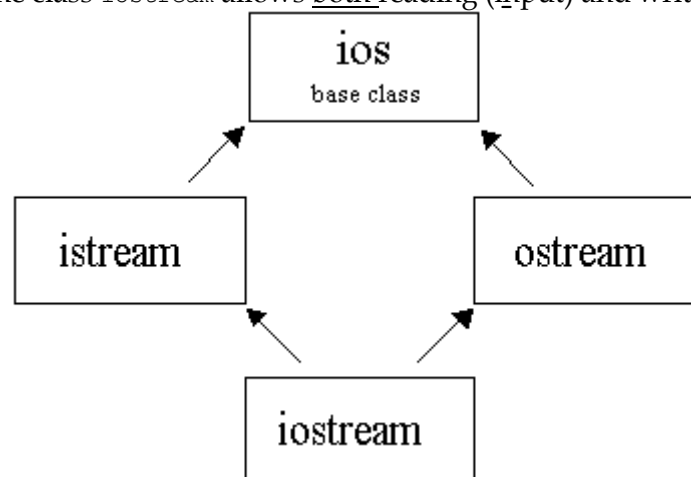


Figure 3: Inheritance hierarchy diagram for stream I/O classes of C++.

Here is an example of a program that makes use of `cin` and `cout` for the keyboard and display monitor. The stream-input operation `getline()` is used to read in exactly one line of text (a sequence of characters ending with the end of line character `'\n'` that the user enters using the keyboard).

```
// Contains declarations of cin and cout.
#include <iostream>
using namespace std;
```

```

int main()
{ const int SIZE = 30; // Maximum number of characters including the
 '\n'
 char yourLine[SIZE];
 cout << "What is your line?" << endl;
 // The ENTER key generates an end of line character.
 // Read chars until user presses ENTER or up to the first SIZE-1
 // characters with any extra characters thrown away and a '\n'
 // character inserted automatically by getline() as the last
 // character put into the array.
 cin.getline(yourLine, SIZE);
 cout << "\nYou said: " << yourLine << endl;
 return 0;
}

```

If we compile the above program into the executable file and then run it, we obtain the following (output from the program is in red, input from the user is in black).

```

What is your line?
My gosh! It works!
You said: My gosh! It works!

```

### Reading from and writing to files using the I/O classes:

To keep data in permanent storage, you can input and output information to files instead of using the keyboard and the display. Conceptually, we have seen that streams can connect the keyboard and monitor to our program. Additionally, an input stream can also connect a file to our program and an output stream, our program to a file. These concepts are illustrated in the following diagrams:

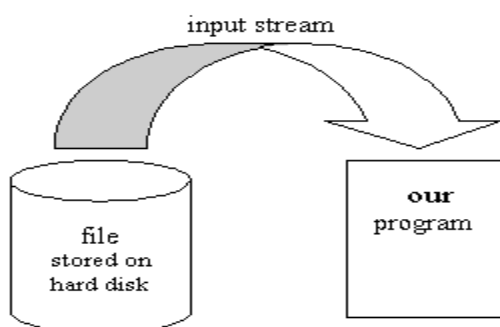


Figure 4: Using a file for the input stream.

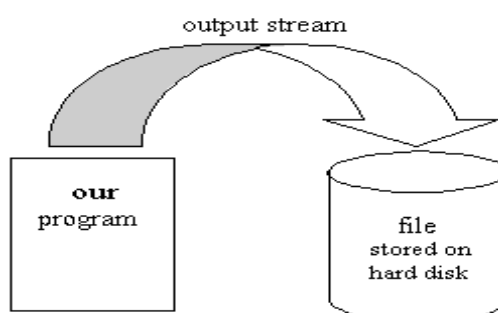


Figure 5: Using a file for the output stream.

C++ implements these file input and output streams using the subclasses `ifstream` and `ofstream`, respectively, where the subclass `ifstream` is derived from the `istream` class and the subclass `ofstream` is derived from the `ostream` class. The `ifstream` and `ofstream` classes inherit all



the stream operations of the `istream` and `ostream` classes, but they also have their own member functions such as `open()` and `close()` and control their relationship to files.

We use the subclass `fstream` to create file stream objects that allow input and output to the same file. Conceptually, this would be represented by a bi-directional "stream" connecting our program to a file. The subclass `fstream` is derived from the `iostream` class using single inheritance.

The file stream inheritance hierarchy we have described above is illustrated by the following diagram.

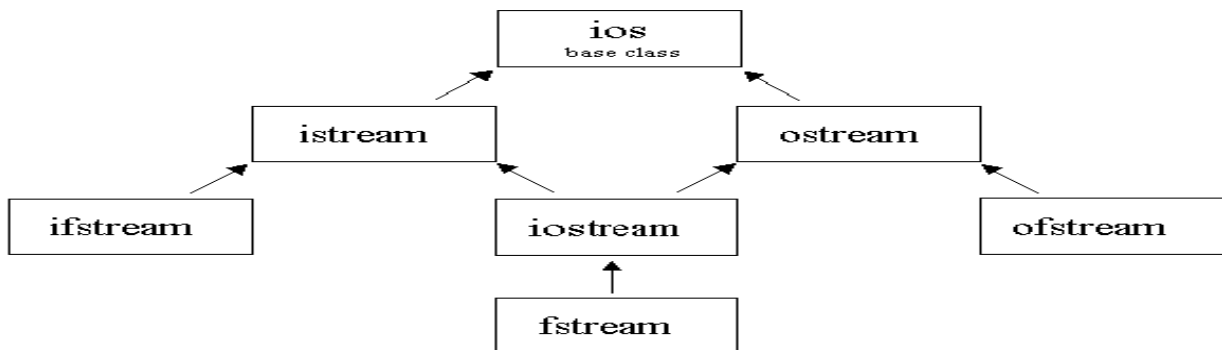
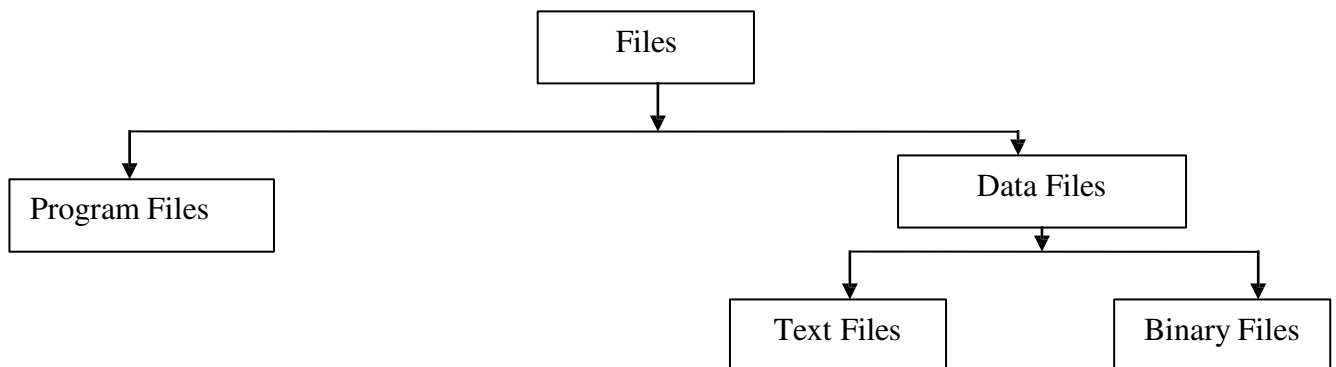


Figure 6: Inheritance hierarchy diagram for stream I/O classes of C++ including file stream classes

Because of inheritance, familiar C++ stream I/O member functions, operators and manipulators, such as `>>` and `<<`, can all be used with file streams. As was noted earlier, you do not need to know the details of how inheritance works to do this lab. For now it is sufficient to know that the file stream classes have all of the input and output operations described here.

### Using File I/O:

This lab will focus on the `ifstream`, `ofstream`, and `fstream` classes. You will modify a driver for the class `FullTimeEmployee` so that test data will be read from an input file and results will be written to an output file. To prepare yourself for this lab, read the following tutorial on File I/O. Files are required to save our data for future use, as Ram is not able to hold our data permanently.



The Language like C/C++ treat everything as a file , these languages treat keyboard , mouse, printer, Hard disk , Floppy disk and all other hardware as a file.

The Basic operation on text/binary files are: Reading/writing, reading and manipulation of data stored on these files. Both type of files needs to be open and close.

### How to open a File:

| Using member function Open()                | Using Constructor                                           |
|---------------------------------------------|-------------------------------------------------------------|
| Syntax<br><code>Filestream object;</code>   | Syntax<br><code>Filestream object("filename", mode);</code> |
| <code>Object.open("filename", mode);</code> | Example                                                     |
| Example                                     |                                                             |

|                                              |                                     |
|----------------------------------------------|-------------------------------------|
| <pre>ifstream fin; fin.open("abc.txt")</pre> | <pre>ifstream fin("abc.txt");</pre> |
|----------------------------------------------|-------------------------------------|

NOTE: (a) Mode are optional and given at the end .

(b) Filename must follow the convention of 8.3 and it" s extension can be anyone

### How to close a file:

All types of files can be closed using **close( )** member function

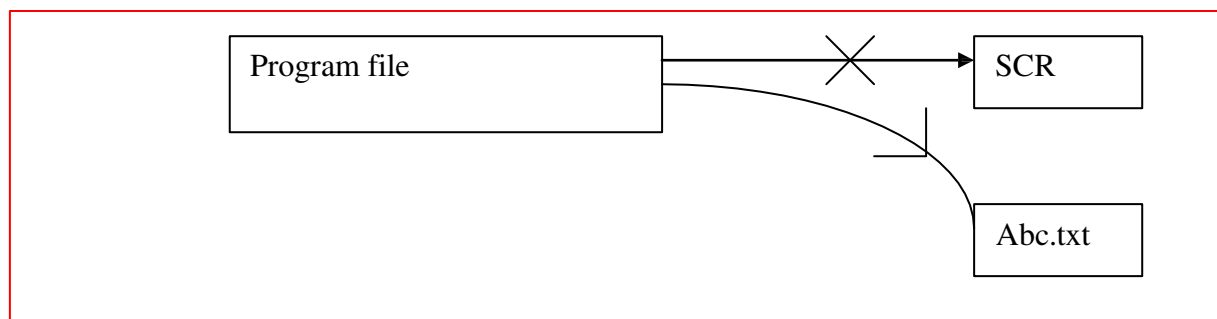
Syntax

```
fileobject.close();
```

Example

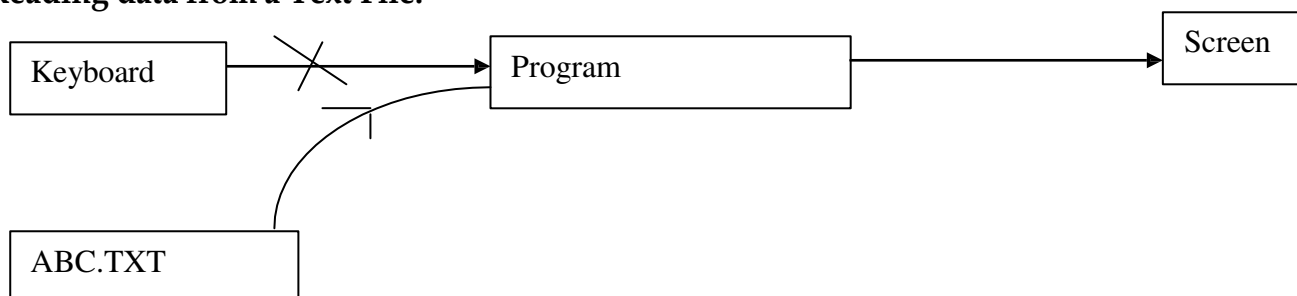
```
fin.close(); // here fin is an object of ifstream class
```

**Objective: To insert some data on a text file**



| Program                                                                                                                                                                                                                                            | ABC.txt file contents                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| <pre>#include&lt;fstream&gt; using namespace std; int main() {     ofstream fout;     fout.open("abc.txt");     fout&lt;&lt;"This is my first program in file handling";     fout&lt;&lt;"\n Hello again";     fout.close();     return 0; }</pre> | <pre>This is my first program in file handling Hello again</pre> |

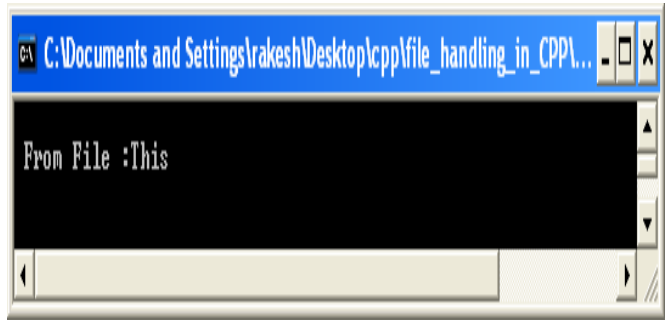
**Reading data from a Text File:**



```

#include<fstream>
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
 ifstream fin;
 char str[80];
 fin.open("abc.txt");
 fin>>str; // read only first
//string from file
 cout<<"\n From File :"<<str;
// as //spaces is treated as
termination point
 getch();
 return 0;
}

```



NOTE : To overcome this problem use `fin.getline(str,79);`

## Detecting END OF FILE

### Using EOF() member function

#### Syntax

```
Filestream_object.eof();
```

#### Example

```

#include<iostream>
#include<fstream>
#include<conio.h>
using namespace std;
int main()
{
 char ch;
 ifstream fin;
 fin.open("abc.txt");
 while (!fin.eof()) // using eof()
 // function
 {
 fin.get(ch);
 cout<<ch;
 }
 fin.close();
 getch();
 return 0;
}

```

### Using filestream object

#### Example

```

// detectting end of file
#include<iostream>
#include<fstream>
#include<conio.h>
using namespace std;
int main()
{
 char ch;
 ifstream fin;
 fin.open("abc.txt");
 while(fin) // file object
 {
 fin.get(ch);
 cout<<ch;
 }
 fin.close();
 getch();
 return 0;
}

```

Example : To read the contents of a text file and display them on the screen.

#### Program ( using getline member function)

```

#include<fstream>
#include<conio.h>
#include<iostream>
using namespace std;
int main()
{
 char str[100];
 ifstream fin;
 fin.open("c:\\abc.txt");
 while(!fin.eof())
 {
 fin.getline(str,99);
 cout<<str;
 }
 fin.close();
 getch();
 return 0;
}

```

#### Program ( using get() member function)

```

#include<fstream>
#include<conio.h>
#include<iostream>
using namespace std;
int main()
{
 char ch;
 ifstream fin;
 fin.open("file6.cpp");
 while(!fin.eof())
 {
 fin.get(ch);
 cout<<ch;
 }
 fin.close();
 getch();
 return 0;
}

```

## Writing/Reading Data in Binary Format

To write and read data in binary format two member functions are available in C++. They are `write()` and `read()`.

**Fileobject.write( (char \*)&object, sizeof(object));**

Syntax for Write( ) member function

**Fileobject.read( (char \*)&object, sizeof(object));**

Syntax for read( ) member function

### Example of write ( ) member function

```
#include<fstream>
#include<iostream>
using namespace std;
struct student
{
 int roll ;
 char name[30];
 char address[60];
};
int main()
{
 student s;
 ofstream fout;
 fout.open("student.dat");
 cout<<"\n Enter Roll Number :";
 cin>>s.roll;
 cout<<"\n Enter Name :";
 cin>>s.name;
 cout<<"\n Enter address :";
 cin>>s.address;
 fout.write((char *) &s, sizeof(student));
 fout.close();
 return 0;
}
```

### To Read data from a binary File using read( ) member function

```
#include<fstream>
#include<iostream>
#include<conio.h>
using namespace std;
struct student
{
 int roll ;
 char name[30];
 char address[60];
};
int main()
{
 student s;
 ifstream fin;
 fin.open("student.dat");
 fin.read((char *) &s, sizeof(student));
 cout<<"\n Roll Number :"<<s.roll;
 cout<<"\n Name : "<<s.name;
 cout<<"\n Address : "<<s.address;
 fin.close();
 getch();
 return 0;
}
```

## Writing Class object in a file

```

#include<fstream>
#include<iostream>
using namespace std;
class student
{
 int roll ;
 char name[30];
 char address[60];
public:
 void read_data(); // member function prototype
 void write_data(); // member function prototype
};
void student::read_data() // member function definition
{
 cout<<"\n Enter Roll :";
 cin>>roll;
 cout<<"\n Student name :";
 cin>>name;
 cout<<"\n Enter Address :";
 cin>>address;
}
void student:: write_data()
{
 cout<<"\n Roll :"<<roll;
 cout<<"\n Name :"<<name;
 cout<<"\n Address :"<<address;
}
int main()
{
 student s;
 ofstream fout;
 fout.open("student.dat");
 s.read_data(); // member function call to get data from KBD
 fout.write((char *)&s,sizeof(student)); // write object in file
 fout.close();
 return 0;
}

```

## Reading Class object from a binary file

```

#include<fstream>
#include<iostream>
#include<conio.h>
using namespace std;
class student
{
 int roll ;
 char name[30];
 char address[60];
public:
 void read_data(); // member function prototype
 void write_data(); // member function prototype
};
void student::read_data() // member function definition
{
 cout<<"\n Enter Roll :";
 cin>>roll;
 cout<<"\n Student name :";
 cin>>name;
 cout<<"\n Enter Address :";
 cin>>address;
}
void student:: write_data()
{
 cout<<"\n Roll :"<<roll;
 cout<<"\n Name :"<<name;
 cout<<"\n Address :"<<address;
}
int main()

```

```

{
 student s;
 ifstream fin;
 fin.open("student.dat");
 fin.read((char *)&s, sizeof(student));
 s.write_data();
 fin.close();
 getch();
 return 0;
}

```

### Some other very important member function:

| Member function name | Explanation                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| seekg()              | Used to move <b>reading</b> pointer forward and backward<br>Syntax<br><b>fileobject.seekg( no_of_bytes,mode);</b><br>Example:<br>(a) fout.seekg(50,ios::cur); // move 50 bytes forward from current position<br>(b) fout.seekg(50,ios::beg); // move 50 bytes forward from current beginning<br>(c) fout.seekg(50,ios::end); // move 50 bytes forward from end . |
| seekp()              | Used to move <b>writing</b> pointer forward and backward<br>Syntax<br><b>fileobject.seekp(no_of_bytes,mode);</b><br>Example:<br>(a) fout.seekp(50,ios::cur); // move 50 bytes forward from current position<br>(b) fout.seekp(50,ios::beg); // move 50 bytes forward from current beginning<br>(c) fout.seekp(50,ios::end); // move 50 bytes forward from end .  |
| tellp()              | It return the distance of <b>writing</b> pointer from the beginning in bytes<br>Syntax<br>Fileobject.tellp( );<br>Example:<br>long n = fout.tellp( );                                                                                                                                                                                                            |
| tellg()              | It return the distance of <b>reading</b> pointer from the beginning in bytes<br>Syntax<br>Fileobject.tellg( );<br>Example:<br>long n = fout.tellg( );                                                                                                                                                                                                            |

### Files MODES:

| File mode      | Explanation                                                                           |
|----------------|---------------------------------------------------------------------------------------|
| ios::in        | Input mode - Default mode with <b>ifstream</b> and files can be read only             |
| ios::out       | Output mode- Default with <b>ofstream</b> and files can be write only                 |
| ios::binary    | Open file as binary                                                                   |
| ios::app       | Preserve previous contents and write data at the end ( move forward only)             |
| ios::ate       | Preserve previous contents and write data at the end.(can move forward and backward ) |
| ios::nodelete  | Do not delete existing file                                                           |
| ios::noreplace | Do not replace file                                                                   |
| ios::nocreate  | Do not create file                                                                    |

**NOTE :** To add more than one mode in a file stream use bitwise OR ( | ) operator

## Difference and Definition

| <b>Text Files</b>                                                                                                        | <b>Binary Files</b>                                                                                                                                                                                 |
|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| In these types of files all the data is firstly converted into their equivalent char and then it is stored in the files. | In these types of files all the data is stored in the binary format as it is stored by the operating system. So no conversion takes place. Hence the processing speed is much more than text files. |

| <b>get() member function</b>                                                                                                                                                       | <b>getline() function</b>                                                                                                                                                                                                                               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get() function is used to read a single char from the input stream in text file<br><b>Syntax</b><br>fileobject.get(char);<br><b>Example:</b><br>fin.get(ch); //fin is file stream. | Getline() function is used to read a string from the input stream in text file.<br><b>Syntax</b><br>fileobject.getline (string, no_of_char,delimiter );<br><b>Example</b><br>fin.getline(str,80); // fin is file stream.<br>NOTE: Delimiter is optional |

### Program to explain the different operation for Project:

```
#include<iostream>
#include<fstream>
#include<conio.h>
using namespace std;
class student
{
 int admno;
 char name[30];
 char address[60];
public:
 void read_data()
 { cout<<"\n Enter Admission No :";
 cin>>admno;
 fflush(stdin);
 cout<<"\n Enter Name :";
 cin.getline(name,29);
 fflush(stdin);
 cout<<"\n Enter Address :";
 cin.getline(address,59);
 }
 void write_data()
 { cout<<"\n\n Admission No:"<<admno;
 cout<<"\n Name :"<<name;
 cout<<"\n Address :"<<address;
 }
 int get_admno()
 {
 return admno;
 }
};
void write_to_file(void)
{
 student s;
 ofstream fout;
 fout.open("student.dat",ios::app);
 s.read_data();
 fout.write((char *)&s,sizeof(student));
 fout.close();
 return;
}
void read_from_file()
{
 student s;
 ifstream fin;
```

```

 fin.open("student.dat");
 while(fin.read((char *)&s,sizeof(student)))
 s.write_data();
 fin.close();
 return;
 }
// function to modify student information
void modify_record(void)
{
 int temp_admno;
 student s;
 ifstream fin;
 ofstream fout;
 fin.open("student.dat");
 fout.open("temp.dat");
 system("cls"); // header file stdlib.h
 cout<<"\n Enter admission No to Modify :";
 cin>>temp_admno;
 while(fin.read((char *)&s,sizeof(student)))
 {
 if (temp_admno==s.get_admno())
 {
 s.read_data();
 }
 fout.write((char *)&s,sizeof(student));
 }
 fin.close();
 fout.close();
 remove("student.dat");
 rename("temp.dat","student.dat");
 return;
}
void modify_alternate_method()
{
 student s;
 int temp_admno;
 fstream file;
 file.open("student.dat",ios::in|ios::out|ios::ate|ios::binary);
 cout<<"\n Enter admno to modify :";
 cin>>temp_admno;
 file.seekg(0); // one method to reach at begining
 // long n = file.tellg(); // find out total no of bytes
 // file.seekg((-1)*n,ios::end); // move backward total no of bytes from end
 while(file.read((char *)&s,sizeof(student)))
 {
 if(temp_admno == s.get_admno())
 {
 s.read_data();
 int n = -1*sizeof(student);
 file.seekp(n,ios::cur);
 file.write((char *)&s,sizeof(student));
 }
 }
 file.close();
 return;
}
void delete_record(void)
{
 int temp_admno;
 student s;
 ifstream fin;
 ofstream fout;
 fin.open("student.dat");
 fout.open("temp.dat");
 system("cls");
 cout<<"\n Enter admission No to Delete :";
 cin>>temp_admno;
 while(fin.read((char *)&s,sizeof(student)))
 {
 if (temp_admno!=s.get_admno())
 fout.write((char *)&s,sizeof(student));
 }
}

```





```

 break;
 case 4: modify_alternate_method();
 break;
 case 5: delete_record();
 break;
 case 6: count_record();
 break;
 case 7: search_record();
 break;
 case 8: break;

 default: cout<<"\n Wrong choice.... Try again";
 getch();
 }
 }while(choice!=8);
 return 0;
}

```

### UNSOLVED PROBLEMS

Problem 1: Observe the program segment carefully and answer the question that follows:

```

class item
{int item_no;
 char item_name[20];
public:
 void enterDetail();
 void showDetail();
 int getItem_no(){ return item_no;}
};
void modify(item x, int y)
{fstream File;
 File.open("item.dat", ios::binary | ios::in | ios::out);
 item i;
 int recordsRead = 0, found = 0;
 while(!found && File.read((char*) &i , sizeof (i)))
 {
 recordsRead++;
 if(i . getItem_no() == y)
 {
 _____//Missing statement
 File.write((char*) &x , sizeof (x));
 found = 1;
 }
 }
 if(! found)
 cout<<"Record for modification does not exist";
 File.close();
 }

```

If the function modify( ) is supposed to modify a record in the file " item.dat ", which item\_no is y, with the values of item x passed as argument, write the appropriate statement for the missing statement using seekp( ) or seekg( ), whichever is needed, in the above code that would write the modified record at its proper place.

Problem 2: Observe the program segment carefully and answer the question that follows:

```

class member
{ int member_no;
 char member_name[20];
public:
 void enterDetail();
 void showDetail();
 int getMember_no(){ return member_no;}
};

```

```

void update(member NEW)
{ fstream File;
 File.open("member.dat", ios::binary|ios::in|ios::out) ;
 member i;
 while(File .read((char*) & i , sizeof (i)))
 {
 if(NEW . getMember_no() = i . getMember_no())
 {
 _____//Missing statement
 File.write((char*) &NEW , sizeof (NEW));
 }
 }
 File.close() ;
 }

```

If the function update( ) is supposed to modify the member\_name field of a record in the file " member.dat" with the values of member NEW passed as argument, write the appropriate statement for the missing statement using seekp( ) or seekg( ), whichever is needed, in the above code that would write the modified record at its proper place.

Problem 3: Given the binary file STUDENT.DAT , containing the records of the following class:

```

class student
{int roll_no;
char name[20];
float percent;
public:
void getData();
void show();
float returnPercent()
{return percent;
}
};

```

Write a function BELOW75( ) in C++ , that would count and display the records of those students whose score is below 75 percent.

Problem 4:

```

class book
{int book_no;
char book_name[20];
float price;
public:
void enter_book_Details()
{
cin>> book_no>> price; gets(book_name);
}
void show_book_Details();
};

```

Assuming a binary file "BOOK.DAT" contains objects belonging to class book, write a user-defined function to add more records to the end of it.

Problem 5: Write a function in C++ to count and display the number of student records stored in the binary file "Student.dat" . Assume that student is a structure and 10 bytes of memory is required to store each student record.

Problem 6: Write a function in C++ to count the number of uppercase alphabets present in a text file "STORY.TXT".

Problem 7: Write a function in C++ to count the number of alphabets present in a text file "XY.TXT".

Problem 8: Write a function in C++ to count and display the number of lines starting with alphabet „A“ in a text file "MYFILE.TXT".

Problem 9: Write a function in C++ to count the number of words present in the text file "MyFile.txt". Assume that each word is separated by a blank space and no blank space appears in the beginning and at the end of the file.

Problem 10: 26 A librarian maintains the record of books in a file named as "STOCK\_BOOK.DAT". Write a function in C++ to delete a record for book\_no 10.

- Problem 11: Given the binary file TELEPHONE.DAT , containing the records of the following class Directory:
- ```

class Directory
{
char name[20];
char address[30];
char areaCode[5];
char phone_no[15];
public:
void register( );
void show( );
int checkCode( char AC[] )
{
return strcmp(areaCode, AC);
}
};

```
- Write a function COPYABC() in C++ , that would copy only those records having areaCode as“123” from TELEPHONE.DAT to TELEBACK.DAT.
- Problem 12: Write a function in C++ to count the number of vowels present in a text file STORY.TXT”.
- Problem 13: Observe the program segment carefully and answer the question that follows:
- ```

class item
{
int item_no;
char item_name[20];
public:
void enterDetail();
void showDetail();
int getItem_no(){ return item_no;}
};
void modify(item x)
{
fstream File;
File.open(“item.dat”, ios::binary|ios::in|ios::out) ;
item i;
while(File .read((char*) & i , sizeof (i))//Statement 1
{
if(x . getItem_no() = = i . getItem_no())
{
File.seekp(File.tellg() – sizeof(i));
File.write((char*) &x , sizeof (x));
}
}

File.close() ;
}

```
- If the function modify( ) modifies a record in the file “ item.dat” with the values of item x passed as argument, rewrite statement 1 in the above code using ios::eof( ) , so as to modify record at its proper place.
- Problem 14: A file named as “STUDENT.DAT” contains the student records, i.e. objects of class student. Assuming that the file is just opened through the object FILE of fstream class, in the required file mode, write the command to position the get pointer to point to fifth record from the beginning.

## : Pointers

### **Pointers**

Introduction, static memory allocation, dynamic memory allocation, declaration and initialization of pointers, pointer arithmetic, arrays and pointers, functions and pointers, structure and pointers, self referential structure, objects and pointers, objects and pointers, this pointer, pointer to pointer

### **Introduction:**

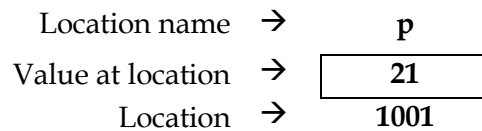
A pointer is a variable which holds a memory address, typically the location of another variable. Any variable declared in a program has two components:

- (i) Address of the variable
- (ii) Value stored in the variable

For example: `int p = 21;`

The above declaration tells the C++ compiler for

- (i) Reservation of space in memory for storing the value.
- (ii) Associating the name `p` with this memory location.
- (iii) Storing the value 21 at this location.



### **Static Memory Allocation:**

When the amount of memory to be allocated is known in advance and memory is allocated during compilation, it is referred to as static memory allocation. For example:

```
int a=10;
```

### **Dynamic Memory Allocation:**

When the amount of memory to be allocated is not known in advance and memory is allocated during execution, it is referred to as dynamic memory allocation. There are two operators `new` and `delete` in C++ for dynamic memory allocation. The operator `new` allocates memory dynamically whereas the operator `delete` is used for deallocation, when the memory is no longer in use.

### **Declaration and Initialization of Pointers:**

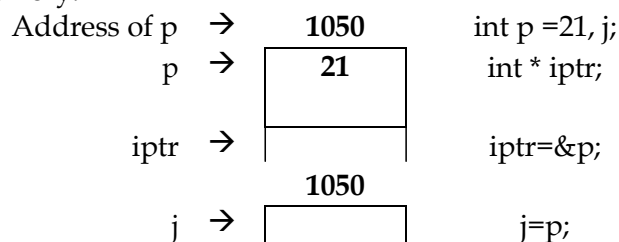
A pointer variable name is preceded by an asterisk (\*) sign. The data type of the pointer should be the same as the data type of the variable to which it will point.

Syntax: `type *var_name;`

Example:

```
int *iptr; //declaration of an integer pointer
int p = 21;
iptr= &p; //iptr stores the address of integer variable p
```

C++ has two unary operators for referencing to the components of any variable. `&` (address operator) returns the *address* of the variable and `*` (indirection operator) returns the *value* stored at any address in the memory.



### **new operator**

Syntax: `pointer variable = new data_type;`

Example:

```
char *cptr;
cptr = new char;
char *captr;
captr = new char[10];
```

delete operator

Syntax: delete pointer variable;

Example:

```
delete cptr;
delete [] captr;
```

C++ has the concept of constant pointer and pointer to a constant. For example:

```
char * const cptr1="Computer Science"; //constant pointer
```

Here the address of cptr1 cannot be modified.

```
int const *iptr=&x; //pointer to a constant
const char * const cptr2="KV"; //pointer to a constant
```

Here the constant value, to which the pointer pointing to, can not be modified.

A NULL pointer is a pointer which indicates that it is not pointing to any valid memory address. For example:

```
int *iptr=NULL;
```

### Pointer Arithmetic:

Only addition and subtraction may be performed on pointers. All the pointers increases and decreases by the length of the data type they point to. Adding 1 to a pointer adds the size of pointer's base type. Let iptr be an integer pointer, currently pointing to memory address 2002. If the int size is 2 bytes, then after the execution of

```
iptr++;
```

iptr will be pointing to 2004.

### Arrays and Pointers:

An array name is equivalent to a pointer pointing to the first element of array. The address of the first byte is called Base Address. In C++ we may have an array of pointers also. If an array name (a pointer actually) is incremented, it points to the next element of the array.

Example: What will be the output of the following program:

```
#include<iostream.h>
void main()
{
int Numbers[] = {2,4,8,10};
int *ptr = Numbers;
for (int C = 0; C<3; C++)
{cout<< *ptr << "@";
ptr++;
}
cout<<endl;
for(C = 0; C<4; C++)
{(*ptr)* = 2;
--ptr;
}
for(C = 0; C<4; C++)
cout<< Numbers [C]<< "#";
cout<<endl;
}
```

**Output:**

```
2@4@8@
4#8#16#20#
```

**Functions and Pointers:**

A function may return a reference or a pointer variable. Pointer to a function can be passed to function, returned from functions.

**Structure and Pointers:**

C++ allows pointers to structures like other data types and these pointers to structures are known as structure pointers.

Syntax: `struct_name *struct_pointer;`

The members of the structure are accessed by using `->` (arrow operator).

Syntax: `struct_pointer->struct_member;`

**Example:** Find the output of the following program:

```
#include<iostream.h>
struct Game
{
 char magic[20];
 int score;
};
void main()
{
 Game M={"Tiger", 500};
 Char * Choice;
 Choice =M.Magic;
 Choice[4]="P";
 Choice[2]="L";
 M.Score+=5;
 cout<< M.Magic<<M.Score<<endl;
 Game N=M;
 agic[0]="A"; Magic[3]="J";
 N.Score-=120;
 Cout<<N.magic<<N.Score<<endl;
}
```

**Output:**

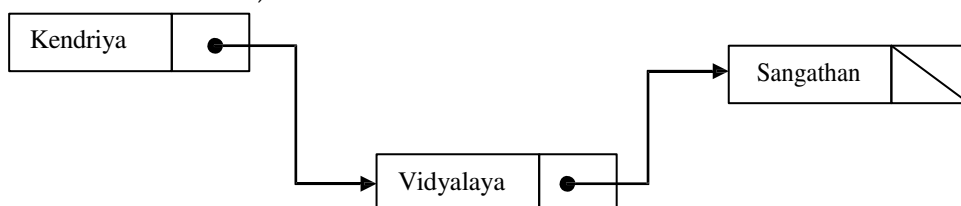
```
TiLeP550
AiLJP430
```

**Self Referential Structures:**

When an element of a structure is declared as a pointer to the structure itself, this type of structure is called self-referential structure. Example:

```
struct node
{
 char data[20];
 node *next;
};
```

- The **node** structure contains both a data member and a pointer to the next node structure (hence, self-referential)



- **next** of first node points to next node
- **next** of last node = NULL

- The data structure itself consists of one or more nodes, "linked" by pointers

### **Objects and Pointers:**

C++ allows us to have pointers to objects known as object pointers.

Syntax: `class_name *object_pointer;`

### **This Pointers:**

While defining a class the space is allocated for member functions only once and separate space is allocated for each object. There exists a serious problem i.e. which object's data member is to be manipulated by any member function.

The "this" pointer is an already created object pointer that points to currently calling object. The this pointer is automatically passed to a member function when it is called.

For Example:

```
#include<iostream.h>
#include<string.h>
class employee
{
 char name[20];
 float salary;
public:
 employee(char *n, float s)
 { strcpy(name,s);
 salary=s;
 }
 employee greater(employee &e)
 { if(e.salary>=salary)
 return &e;
 else
 return this;
 }
 void display()
 {
 cout<<"\nName:"<<name;
 cout<<"\nSalary:"<<salary;
 }
};
void main()
{employee e1("ABC",10000), e2("PQR",20000), e3("XYZ",5000);
 employee *emp;
 emp=e1.greater(e3); e1-
 >display();
 emp=e2.greater(e3); e2-
 >display();
}
```

### **Output:**

```
Name:ABC
Salary:10000
Name:PQR
Salray:20000
```

### **Pointers to Pointers:**

We can define pointer pointing to another pointer that points to the target value.

Syntax: `int **ptr_to_ptr;`

In the case of pointer to a pointer, the first pointer contains the address of the second pointer, which points to the objects that contains the value desired.

**Example:** Give the output of the following program segment (assume all required header files are included in the program):

```
char *NAME= "a ProFile";
```



```

for(int x=0;x<strlen(NAME);x++)
 if(islower(NAME[x]))
 NAME[x]=toupper(NAME[x]);
 else
 if(isupper(NAME[x]))
 if(x%2!=0)
 NAME[x]=tolower(NAME[x-1]);
 else
 NAME[x]--;
 Cout<<NAME<<endl;

```

**Output:**

A OROoILE

**Example:** Give the output of the following program segment (assume all required header files are included in the program):

```

char *s= "STUDY";
for(int x=0; x<strlen(s);x++)
 {for(int y=0; y<=x; y++)
 cout<<s[y];
 cout<<endl;
 }

```

**Output:**

S  
ST  
STU  
STUD  
STUDY

**Example:** Find the output of the following program:

```

#include<iostream.h>
void main()
{
 int X[] = {10,25,30,55,110};
 int *p = X;
 while(*p< 110)
 {
 if (*p%3!=0)
 *p =*p + 1;
 else
 *p =*p + 2;
 p++;
 }
 for(int i=4;i>=1;i--)
 { cout<<X[i]<<"@";
 if (i%3==0) cout<<endl;
 }
 cout<<X[0]*3<<endl;
}

```

**Output:**

110@56@  
32@26@33

## UNSOLVED PROBLEMS

- Problem 1:** Give the output of the following program:
- ```
#include<iostream.h>
void main( )
{
    int a =32, *ptr = &a;
    char ch = „A“, &cho = ch;
    cho += a; *ptr += ch;
    cout << a << " " << ch << endl;
}
```
- Problem 2:** Give the output of the following program (Assuming all required header files are included in the program):
- ```
void main()
{
 int array []={ 2, 3, 4, 5};
 int *arptr = array;
 int value =*arptr;
 value = *arptr++;
 cout << value <<“\t”;
 value = *arptr;
 cout << value <<“\t”;
 value = * ++arptr;
}
```
- Problem 3:** Find the output of the following program:
- ```
#include <iostream.h>
#include <string.h>
class state
{
    char *state_name;
    int size;
public:
    state()
    { size=0;
      state_name = new char [size+1];
    }
    state (char *s)
    {
        size = strlen(s);
        state_name = new char[size + 1];
        strcpy(state_name, s);
    }
    void display( )
    { cout<<state_name<<endl;
    }
    void Replace (state & a, state & b)
    { size = a.size + b.size;
      delete state_name;
      state_name = new char[size + 1];
      strcpy(state_name, a.state_name);
      strcat(state_name, b.state_name);
    }
};
void main( )
{
    char * temp = "Delhi";
    state state1(temp),state2("Mumbai"),state3("Nagpur"), S1, S2;
    S1.Replace(state1, state2);
    S2.Replae(S1, State3);
    S1.display();
    S2.display();
}
```
- Problem 4:** What will be the output of the following program:
- ```
#include<iostream.h>
#include<ctype.h>
```

```

#include<conio.h>
#include<string.h>
void changestring(char text[], int &counter)
{
 char *ptr = text;
 int length=strlen(text); for(;counter<length-
2;counter+=2,ptr++)
 {
 (ptr+counter) = toupper((ptr+counter));
 }
}
void main()
{
 clrscr();
 int position = 0;
 char message[]= "Mouse Fun";
 changestring (Message, position);
 cout<<message<< "@" <<position;
}

```

**Problem 5:** Find the output of the following program:

```

#include<iostream.h>
#include<string.h>
class country
{ char *country_name;
 int length;
public:..
 country ()
 {length =0; country_name=new char [length+1];}
 country (char *s)
 {length = strlen(s);
 country_name=new char [length +1];
 strcpy (country_name, s);
 }
 void display ()
 {cout<< country_name <<endl;
 }
 void Replace (country & a, country & b)
 {length a.length + b.length;
 delete country_name;
 country_name=new char [length + 1];
 strcpy (country_ name, a.country_name);
 strcat (country_name, b.country name);
 }
};
void main ()
{char * temp = "India";
 Country country1(temp),country2("Nepal"),country3("China"), S1,S2;
 S1.Replace (country1, country2);
 S2.Replace (S1,country3);
 S1.display();
 S2.display ();
}

```

**Problem 6:** What will be the output of the following program

```

#include<iostream.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
void changestring(char text[], int &counter)
{char *ptr = text;
 int length=strlen(text); for(;counter<length-
2;counter+=2,ptr++)
 {*(ptr+counter) = toupper(*(ptr+counter));
 }
}

```

```
void main()
{clrscr();
 int position = 0;
 char message[] = "Mouse Fun";
 changestring (Message, position);
 cout<<message<< "@" <<position;
}
```

## Unit 2: Data Structures

### : Arrays

#### ARRAYS

Introduction to data structure, Arrays, One Dimensional Array, Basic operations on 1-D array: traversal, Searching- linear, binary search, insertion, deletion, sorting- insertion, selection, bubble, merge sort, two dimensional array, implementation of 2-D array in memory- row major, column major, basic operations on 2-D array.

#### Introduction to Data Structure:

**Data Type:** A data type refers to a named group of data which share similar properties or characteristics and which have common behaviour among them.

**Data Structure:** A data structure is a named group of data of different data types which can be processed as a single unit. In other words a data structure is a logical method of representing data in memory using simple and complex data types provided by the language.

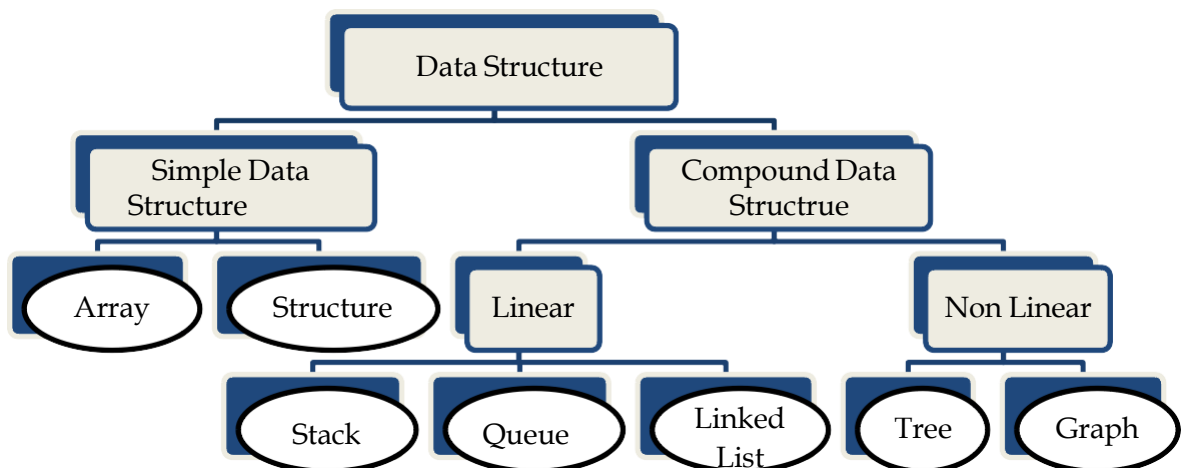
Organized collection of data is called data structure.

**Data Structure = Organized Data + Allowed Operations**

**Data type = Data values + Operations**

The data structure can be classified into following two types:

1. **Simple Data Structure:** These are build from fundamental data types i.e. int, float, char etc.  
Example: Array, Structure
2. **Compound Data Structure:** Simple data structure can be combined in various ways to form more complex structure called compound data structures. It is of two types:
  - (i) **Linear Data Structure:** A data structure is said to be linear if its elements or items are stored sequentially. Example: Stack, Queue, Linked List
  - (ii) **Non-Linear Data Structure:** These are multilevel data structure, in which elements are not stored sequentially. Example: Stack, Queue, Linked List.



#### Arrays:

It is a collection of similar type of data elements. Its elements are allocated contiguous memory. Arrays are of different types:

- (i) One-dimensional array
- (ii) Multi-dimensional array

**One Dimensional Array:**

The simplest form of an array is one-dimensional array. The array itself is given a name and its elements are referred to by their subscripts. The notation of array can be:

Array\_name[lower bound L, upper bound U]

In C++ , its like

Arrayname[size]

where size specifies the number of elements in the array and the subscript (index) value ranges from 0 through size-1.

Length of the array = U- L + 1

Address of element with subscript I = B + S\*(I-L)

where,

B: Base Address (Address of the very first element)

S: Size of an array element

L: Lower Bound of array

**Basic Operations on 1-D Array:**

1. **Traversal:** Visiting each element (from start to end) is called traversal.

```
//Program to illustrate the concept of traversal in an array
#include<iostream.h>
#include<conio.h>
void main()
{int arr[10],n,i;
 clrscr();
 cout<<"\n Enter the number of elements:";
 cin>>n;
 cout<<"\n Enter " << n <<" elements:";
 for(i=0; i<n; i++)
 cin>>arr[i];
 cout<<"\n Entered array is:\n";
 for(i=0; i<n; i++)
 cout<<arr[i]<<" ";
 getch();
}
```

2. **Searching:** searching in a 1-D array can be done in following two ways:

- (i) Linear Search
- (ii) Binary search

**(i)Linear Search:** In linear search (sequential search), each element of the array is compared with given item to be searched for, one by one. This searching technique can work for both unsorted and sorted array.

```
//Linear Search in 1-D unsorted array
#include<iostream.h>
#include<conio.h>
int linear_search(int [],int,int);
void main()
{ int A[20],N, index,i,ITEM;
 clrscr();
 cout<<"Enter number of elements:";
 cin>>N;
 cout<<,"\nEnter the elements:";
 for(int i=0; i<N;i++)
 cin>>A[i];
 cout<<"\n Enter element to be searched:";
 cin>>ITEM;
 index = linear_search(A,N,ITEM);
 if(index==-1)
 cout<<"\n Element not Found:";
```

```

else
 cout <<" "\n Element found at Index "<<index
 <<"and at Position: "<<index+1;
 getch();
}
int linear_search(int a[],int n, int item)
{
 for(int i=0; i<n; i++)
 { if(a[i]==item)
 return i;
 }
return -1;
}

```

**(ii) Binary Search:** The binary search technique can work only for sorted arrays. To search an element say ITEM in a sorted array (suppose in ascending order), the ITEM is compared with the middle element. If the ITEM is greater than the middle element, second half of the array becomes new segment to be scanned, if the ITEM is less than the middle element, first half becomes the new segment to be scanned. The same process is repeated until the ITEM is found or the segment is reduced to the single element and still ITEM is not found.

```

//Binary Search in 1-D sorted array
#include<iostream.h>
#include<conio.h>
int binary_search(int [],int,int);
void main()
{ int A[20],N, index,i,ITEM;
 clrscr();
 cout<<"Enter number of elements:";
 cin>>N;
 cout<<"\nEnter the elements:";
 for(int i=0; i<N;i++)
 cin>>A[i];
 cout<<"\n Enter element to be searched:";
 cin>>ITEM;
 index = binary_search(A,N,ITEM);
 if(index==-1)
 cout<<"\n Element not Found:";
 else
 cout <<" "\n Element found at Index "<<index
 <<"and at Position: "<<index+1;
 getch();
}
int binary_search(int a[],int n, int item)
{int beg,mid,last;
 beg=0;
 last=n-1;
 while(beg<=last)
 { mid = (beg + last)/2;
 if(item==a[mid])
 return mid;
 else if(item>a[mid])
 beg=mid+1;
 else last=mid-
 1;
 }
return -1;
}

```

- 3. Insertion:** Insertion of new element can be done at a specific position and if the array is sorted insertion of the element will be at appropriate place. Insertion is not possible if the

array is already full which is called "OVERFLOW" but replacement of an existing element is possible.

```
//function for inserting an element in an array at a specific position
void insert(int a[],int n, int index,int item)
{ for(int i=n-1; i>=index;i--)
 a[i+1]=a[i];
 a[index]=item;
}
```

**Note:** Check the condition for overflow in main() function

```
//function for inserting an element in a sorted array
void insert(int a[],int n, int item)
{int i,pos;
 if(item>=a[n-1])
 a[n]=item;
 else
 {pos=0;
 While(a[pos]<=item)
 pos++;
 for(i=n-1; i>=pos; i--)
 a[i+1]=a[i];
 a[pos]=item;
}
```

**4. Deletion:** Deletion of an element means its removal from the array. Deletion may not be possible if the element does not exist. Deletion can be done in any one of the following ways:

- (i) Deletion of an element from a specific position
- (ii) Deletion of an element from an unsorted array
- (iii) Deletion of an element from a sorted array.

```
//function for deleting an element in a sorted array
int Del_element(int a[],int n, int item)
{ int i,pos=0;
 If(item<a[0]||item>a[n-1])
 return -1;
 while(a[pos]<=item)
 Pos++;
 If(a[pos]==item)
 {for(i=pos+1; i<n; i++) a[i-
 1]=a[i];
 a[n-1]=0;
 return(pos);
}
return -1;
}
```

**Note:** Check the condition for underflow in main() function

**5. Sorting:** Sorting means arranging the elements in some specific order, i.e. either ascending or descending order. The various sorting techniques available are:

- (i) **Insertion Sort:** Initially, the first element is assumed to be sorted. In the first pass, 2<sup>nd</sup> element is inserted into its proper place in the sorted part of the array. Similarly in the next pass, the 3<sup>rd</sup> element is placed and so on. Given below is the insertion sort for ascending order.

|                            |    |    |    |    |    |    |
|----------------------------|----|----|----|----|----|----|
| <b>Array at beginning:</b> | 42 | 29 | 74 | 11 | 65 | 58 |
| After pass 1               | 29 | 42 | 74 | 11 | 65 | 58 |
| After pass2                | 29 | 42 | 74 | 11 | 65 | 58 |



|                     |           |           |           |           |           |           |
|---------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>After pass 3</b> | 11        | 29        | 42        | 74        | 65        | 58        |
| <b>After pass 4</b> | 11        | 29        | 42        | 65        | 74        | 58        |
| <b>After pass 5</b> | 11        | 29        | 42        | 58        | 65        | 74        |
| <b>Sorted Array</b> | <b>11</b> | <b>29</b> | <b>42</b> | <b>58</b> | <b>65</b> | <b>74</b> |

```
//function for Insertion Sort
void InsertionSort(int a[],int n)
{ int i,j,temp;
 for(i=1;i<n;i++)
 { temp=a[i];
 j=i-1;
 while(temp<a[j]&&j>=0)
 {a[j+1]=a[j];
 j--;
 }
 a[j+1]=temp;
 cout<<"\n After Pass "<<i ;
 for(k=0;k<n;k++)
 cout<<a[k];
 }
}
```

- (ii) **Selection Sort:** The element with the smallest value (if found) is swapped with the first element. As a result of this interchange, the smallest element is placed in the 1<sup>st</sup> position of the array. In the second pass, second smallest element is searched and swapped with second element and so on. Given below is the selection sort for ascending order.

|                            |           |           |           |           |           |           |
|----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>Array at beginning:</b> | 42        | 29        | 74        | 11        | 65        | 58        |
| <b>After pass 1</b>        | <b>11</b> | 29        | 74        | <b>42</b> | 65        | 58        |
| <b>After pass2</b>         | 11        | <b>29</b> | 74        | 42        | 65        | 58        |
| <b>After pass 3</b>        | 11        | 29        | <b>42</b> | <b>74</b> | 65        | 58        |
| <b>After pass 4</b>        | 11        | 29        | 42        | <b>58</b> | 65        | <b>74</b> |
| <b>After pass 5</b>        | 11        | 29        | 42        | 58        | <b>65</b> | 74        |
| <b>Sorted Array</b>        | <b>11</b> | <b>29</b> | <b>42</b> | <b>58</b> | <b>65</b> | <b>74</b> |

```
//function for Selection Sort
void SelectionSort(int a[],int n)
{ int i,small,pos,temp;
 for(i=0;i<n;i++)
 { small=a[i];
 pos=i;
 for(j=i+1;j<n;j++)
 {if(a[j]<small)
 {small=a[j];
 Pos=j;}}
 }
 temp=a[i];
 a[i]=a[pos];
 a[pos]=temp;
 cout<<"\n After Pass "<<i+1 ;
 for(j=0;j<n;j++)
 cout<<a[j];
}
```

- (iii) **Bubble Sort:** In this technique, two adjacent values are compared and they are exchanged if not in proper order. In every pass, the larger element settles at its appropriate position in the bottom. Given below is the bubble sort for ascending order.

|                            |           |           |           |           |           |           |
|----------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>Array at beginning:</b> | 42        | 29        | 74        | 11        | 65        | 58        |
| After pass 1               | 29        | 42        | 11        | 65        | 58        | <b>74</b> |
| After pass2                | 29        | 11        | 42        | 58        | <b>65</b> | 74        |
| After pass 3               | 11        | 29        | 42        | <b>58</b> | 65        | 74        |
| After pass 4               | 11        | 29        | <b>42</b> | 58        | 65        | 74        |
| After pass 5               | 11        | <b>29</b> | 42        | 58        | 65        | 74        |
| <b>Sorted Array</b>        | <b>11</b> | <b>29</b> | <b>42</b> | <b>58</b> | <b>65</b> | <b>74</b> |

```
//function for Bubble Sort
void BubbleSort(int a[],int n)
{ int temp;
 for(int i=0;i<n;i++)
 { for(int j=0;j<n-1;j++)
 {if(a[j]>a[j+1])
 {temp=a[j];
 A[j]=a[j+1];
 A[j+1]=temp;
 }
 }
 }
 cout<<"\n After Pass "<<i+1 ;
 for(int k=0;k<n;k++)
 cout<<a[k];
}
```

- (iv) **Merge Sort:** Merging is the process of combining two or more sorted arrays into another array which is also sorted. In merge sort the array is sorted while merging.

Suppose we have to merge array A and array B. The first element of array A is compared with the first element of array B. If the first element of array A is smaller than the first element of array B, the element from array A is moved to the new array C. The subscript of array A is now increased since the first element is now set and we move on.

If the element from array B should be smaller, it is moved to the new array C. The subscript of array B is increased. This process of comparing the elements in the two arrays continues until either array A or array B is empty. When one array is empty, any elements remaining in the other (non-empty) array are "pushed" into the end of array C and the merge is complete.

```
//function for Merge Sort
void MergeSort(int a[],int b[],int c[],int m,int n)
{ int i=0,j=0,k=0,r;
 while(i<m &&j<n)
 { if(a[i]<=b[j])
 {c[k]=a[i];
 i++;
 }
 else
 {c[k]=b[j];
 j++;
 }
 }
}
```

```

}
k++;
}
if (i==m)
{ for (r=j; r<n; r++)
 { c[k]=b[r];
 k++;
 }
}
else
{ for (r=i; r<m; r++)
 { c[k]=a[r];
 k++;
 }
}
}
}
}

```

### Two Dimensional Array:

A two dimensional array is an array in which each element is itself an array. Two dimensional arrays are called matrices. A matrix resembles a table with rows and columns. In C++, 2-D array notation is

Arrayname[Row][Column]

No of elements = Row \* Column

|      | columns |        |        |        |        |
|------|---------|--------|--------|--------|--------|
| rows | [0][0]  | [0][1] | [0][2] | [0][3] | [0][4] |
|      | [1][0]  | [1][1] | [1][2] | [1][3] | [1][4] |
|      | [2][0]  | [2][1] | [2][2] | [2][3] | [2][4] |
|      | [3][0]  | [3][1] | [3][2] | [3][3] | [3][4] |

### Implementation of 2-D array in Memory:

Elements of a 2-D arrays in memory are allotted contiguous locations. The array elements are stored linearly using one of the following two methods:

1. **Row Major:** Using this method a 2-D array is stored with all the elements of first row in sequence followed by elements of second row and so on.

$$\text{Address of the [I,J]th element} = B + S * [(I-L_r) * N + (J-L_c)]$$

where

B: Base Address (address of very first element)

S: Size of an array element

L<sub>r</sub> : Lower bound of row (first row number)

L<sub>c</sub> : Lower bound of column (first column number)

N : No of columns

**Example:** An array mat[15][30] is stored in the memory with each element requiring 8 bytes of storage. If the base address of V is 5300, find out memory location of mat[8][12], if the array is stored along the row.

**Answer:**

Base address B= 5300

Size of elements S=8 bytes

Number of rows M= 15

Number of columns N=30

Lower bound of row L<sub>r</sub> =0

Lower bound of column  $L_c = 0$

In row major implementation

Address of the  $[I, J]^{\text{th}}$  element =  $B + S * [(I - L_r) * N + (J - L_c)]$

Address of mat  $[8][12] = 5300 + 8 * [(8 - 0) * 30 + (12 - 0)]$   
 $= 5300 + 8 * [8 * 30 + 12]$   
 $= 5300 + 8 * [240 + 12]$   
 $= 5300 + 8 * 252$   
 $= 5300 + 2016$   
 $= 7316$

- 2. Column Major:** Using this method a 2-D array is stored with all the elements of first column in sequence followed by elements of second column and so on.

**Address of the  $[I, J]^{\text{th}}$  element =  $B + S * [(I - L_r) + (J - L_c) * M]$**

where

B: Base Address (address of very first element)

S: Size of an array element

$L_r$ : Lower bound of row (first row number)

$L_c$ : Lower bound of column (first column number)

M: No of rows

**Example:** An array  $A[15][35]$  is stored in the memory along with column with each of its elements occupying 8 bytes of storage. Find out the base address and address of an element  $A[2][5]$ , if the location  $A[5][10]$  is stored at the address 4000.

**Answer:**

Let the Base address be B

Size of elements  $S = 8$  bytes

Number of rows  $M = 15$

Number of columns  $N = 35$

Lower bound of row  $L_r = 0$

Lower bound of column  $L_c = 0$

In column major implementation

Address of the  $[I, J]^{\text{th}}$  element =  $B + S * [(I - L_r) + (J - L_c) * M]$

Address of mat  $[5][10] = B + 8 * [(5 - 0) + (10 - 0) * 15]$

4000 =  $B + 8 * [5 + 10 * 15]$

4000 =  $B + 1240$

B =  $4000 - 1240$

B = 2760

Address of mat  $[2][5] = 2760 + 8 * [(2 - 0) + (5 - 0) * 15]$

=  $2760 + 8 * [2 + 5 * 15]$

=  $2760 + 616$

= 3376

## Basic Operations on 2-D Array:

- 1. Traversal:** Visiting each element (from start to end) is called traversal.

```
//Program to illustrate the concept of traversal in a 2-D array
#include<iostream.h>
#include<conio.h>
void main()
{int arr[10][10], m, n, i, j;
 clrscr();
 cout<<"\n Enter the number of rows:";
 cin>>m;
 cout<<"\n Enter the number of columns:";
 cin>>n;
```

```

cout<<"\n Enter " << m*n <<" elements:";
for(i=0; i<m i++)
 for(j=0; j<n; j++)
 cin>>arr[i][j];
cout<<"\n Entered array is:\n";
for(i=0; i<m i++)
 { cout<<"\n";
 for(j=0; j<n; j++)
 cout<<arr[i][j]<<" ";
 }
getch();
}

```

- 2. Finding Sum/Difference of two M x N matrices:** The sum/difference of two M x N matrices can be obtained by adding/subtracting the corresponding elements of the two arrays.
- 3. Interchanging row and column elements:** The elements of row and column can be interchanged. The resultant matrix is called transpose of the given matrix.
- 4. Product of two matrices:** The condition of matrix multiplication is that the number of columns of first matrix must be equal to number of rows of the second matrix. The resultant matrix will have size equal to number of rows of first matrix \* number of columns of second matrix.

**Example:** Write a function in C++ which accepts an integer array and its size as arguments / parameters and assign the elements into a two dimensional array of integers in the following format :

|                                                                                                                                                                    |                                                                                                       |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <p>If the array is 1, 2,3,4,5,6<br/>The resultant 2D array is given below</p> <pre> 1 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 0 0 1 2 3 0 0 0 1 2 0 0 0 0 1 0 0 0 0 0 </pre> | <p>If the array is 1,2,3<br/>The resultant 2D array is given below</p> <pre> 1 2 3 1 2 0 1 0 0 </pre> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|

**Solution :**

```

void func(int arr[], int size)
{
 int a2[20][20], i, j;
 for (i=0;i<size; i++)
 {
 for (j=0;j<size;j++)
 {
 if ((i+j) >=size)
 a2[i][j]=0;
 else
 a2[i][j]= arr[j];
 cout<<a2[i][j]<<" ";
 }
 Cout<<"\n";
 }
}

```

**Example:** Write a program in c++ which accepts a 2D array of integers and its size as arguments and displays the elements which lies on diagonals.

[ Assuming the 2D array to be a square matrix with odd dimensions , i.e 3x3, 5x5,7x7, etc ]

Example if the array content is

```

5 4 3
6 7 8
1 2 9

```

Output through the function should be

Diagonal one : 5 7 9

Diagonal two : 3 7 1 .

**Solution:**

```
// Function to display the elements which lie on diagonals
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
const M = 10;
const N = 10;
void display_diagonals(int MATRIX[M][N], int r, int c)
{
 clrscr();
 // Finding the diagonal from left index to right
 cout << "Diagonal One : ";
 for(int i=0; i<r; i++)
 for(int j=0; j<c; j++)
 {
 cout << MATRIX[i][j] << " ";
 i++;
 }
 cout << endl;
 // Finding the diagonal from right index to left
 cout << "Diagonal Two : ";
 for(i=0; i<=r; i++)
 {
 for(int j=c-1; j>=0; j--)
 {
 cout << MATRIX[i][j] << " ";
 i++;
 }
 }
 getch();
}
void main()
{
 int MATRIX[M][N];
 int i, j;
 int r, c;
 cout << "Enter total no. of rows: ";
 cin >> r;
 cout << "Enter total no. of columns: ";
 cin >> c;
 if ((r == c) && ((r%2==1) && (c%2==1)))
 {
 cout << "Input steps";
 cout << "\n\nEnter the element in the array\n";
 for(i=0; i<r; i++)
 for(j=0; j<c; j++)
 {
 cin >> MATRIX[i][j];
 }
 }
 else
 return;
 display_diagonals(MATRIX, r, c);
}
```

### UNSOLVED PROBLEMS

**Problem 1:** Write a function to search an element using Binary Search.

**Problem 2:** Write a function in C++ which accepts an integer array and its size as arguments and swaps the elements of every even location with its odd location

eg., if the array initially contains

2, 4, 1, 6, 5, 7, 9, 2, 3, 10

then it should contain

4, 2, 6, 1, 7, 5, 2, 9, 10, 3

- Problem 3: Write a function in C++ to combine the contents of two equi-sized arrays A and B by computing their corresponding elements with the formula  $2 * A[i] + 3 * B[i]$ , where value I varies from 0 to N-1 and transfer the resultant content in the third same sized array.
- Problem 4: Write a function in C++ to merge the contents of two sorted arrays A and B, into the third array C. Assume array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending.
- Problem 5: An array MAT[30][10] is stored in the memory row wise with each element occupying 8 bytes of memory. Find out the base address and the address of the element MAT[15][5], if the location of MAT[5][7] is stored at the address 3000.
- Problem 6: An array X[15][10] is stored in memory with each element requiring 2 bytes of storage. If the base address of array is 2000, calculate the location of X [7][8] when the array is stored by (1) row major order (2) column major order.
- Problem 7: An array ARR[15][35] is stored in the memory along the column with each of its elements occupying 8 bytes. Find out the base address and the address of an element ARR[2][5] , if the location is stored at the address 4000
- Problem 8: Write a function in c++ which accepts a 2D array of integers, number of rows and number of columns as arguments and assign the elements which are divisible by 3 or 5 into a one dimensional array of integers.

If the 2D array is

$$\begin{bmatrix} 12 & 3 & 9 & 14 \\ 24 & 25 & 16 & 31 \\ 19 & 32 & 45 & 27 \\ 11 & 5 & 28 & 18 \end{bmatrix}$$

The resultant 1D arrays is 12, 3, 9, 24, 25, 45, 9, 5, 18

- Problem 9: Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of the middle row and the elements of middle column.

Example if the array content is

3 5 4

7 6 9

2 1 8

Output through the function should be:

Middle row: 769

Middle column: 5 6 1

- Problem 10: Write a user defined function named upperhalf( ) which takes a 2D array A, with size n rows and n columns as arguments and print the upper half of the matrix

1 2 3

1 2 3

6 7 8

7 8

2 3 4

4

## : Stack

### STACK

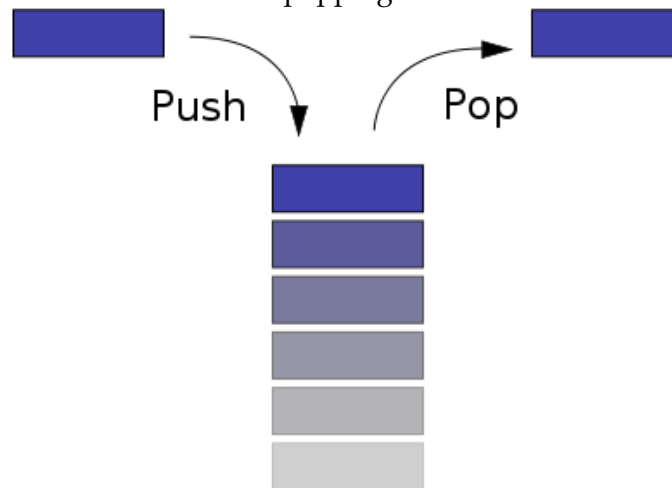
Introduction to linked list, stack, array implementation of stack, linked list implementation of stack, covering INFIX to POSTFIX notation, Evaluation of a POSTFIX expression

### Introduction:

The term list means a linear collection of elements. Array is an example of linear lists. A linked list is a collection of data elements, called nodes pointing to the next nodes by means of pointers. Each node is divided into two parts DATA (info) and LINK (next). It is a dynamic data structure as it can grow or shrink.

### Stack:

It is a list in which insertion or deletion of an element is allowed only at one end. The rule used for a stack is to always remove the item that has been in the collection the *least* amount of time. This policy is known as *last-in first-out* or LIFO. The most accessible element denotes the top and least accessible element denotes the bottom of the stack. An insertion in a stack is called pushing and a deletion from a stack is called popping.



### Array Implementation of Stack:

Array is a static data structure. So the space is allocated according to the maximum number of elements present at that point.

Pushing an element in the stack is possible through top only. In case the array is full and no new element can be accommodated, it is called STACK-FULL. This condition is called OVERFLOW.

Popping i.e. deletion of an element takes place from the top. In case the last element is popped, the stack becomes empty. If one tries to delete an element from an empty stack, this is called UNDERFLOW.

```
//PUSH and POP operations on a stack-array implementation
#include<iostream.h>
#include<conio.h>
#include<process.h>
define SIZE 20
void PUSH(int stack[],int item, int &top)
{if(top==size-1)
 {cout<<"\n OVERFLOW";
 exit(1);
 }
else
 { top++;
 stack[top]=item;
 }
}
```



```

}
int POP(int stack[], int &top)
{ if(top== -1)
 { cout<<"\n UNDERFLOW";
 Exit(1);
 }
else
{ int ret=stack[top];
 top--;
}
return (ret);
}

void DISPLAY(int stack[], int top)
{ if (top== -1)
 cout<<"Stack empty";
 else
 {cout<<stack[top]<<"<--" <<endl;
 for(int i=top-1;i>=0;i--)
 cout<<stack[i]<<endl;
 }
}

void main()
{int stack[size],item, top=-1, res;
 char ch="y";
 clrscr();
while(ch=="y" || ch=="Y")
{cout<<"\n Enter the ITEM for insertion:";
 cin>>item;
 PUSH(stack,item,top);
 cout<<"\n The stack is :\n";
 DISPLAY(stack,top);
 cout<<"\n Want to insert more elements? (y/n)";
 cin>>ch;
}
cout<<"\n Deletion of elements:\n";
ch="y";
while(ch=="y" || ch=="Y")
{
 res=POP(stack,top);
 cout<<"\n Deleted element is:"<< res;
 cout<<"\n The stack is :\n";
 DISPLAY(stack,top);
 cout<<"\n Want to delete more elements? (y/n)";
 cin>>ch;
}
getch();
}

```

### **Linked List Implementation of Stack:**

A linked list is a dynamic data structure. So the space requirement is not predetermined. The linked-stack is created after getting a node for the ITEM to be inserted. TOP points to the newly inserted node. The dynamic allocation of memory is done by using new operator as shown below:

```

Struct node
{ int info;
 Node *next;
}*newptr;

```

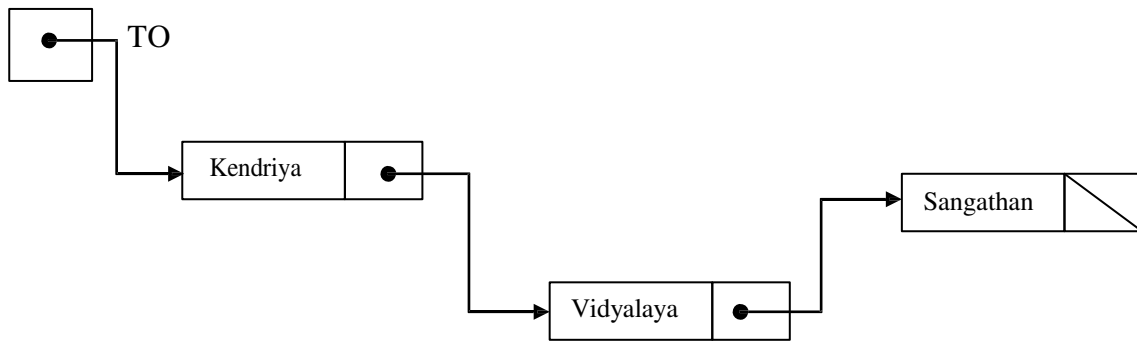
Now, the following statement creates a node dynamically

```

newptr =new node;
if (newptr==NULL)
 cout<<"stack Underflow";

```

Pushing can only occur at the top, TOP gets modified every time.



Popping also requires modification of TOP. Top is made to point to the next node in the sequence. Deallocation of memory is done by the operate delete.

```
delete ptr;
```

**//Program illustrating basic operation of add stack, delete stack**

//and shows stack using linked list.

```

#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
struct node
{
 int data;
 node *link;
};
node *push(node *top, int val); // Add stack
node *pop(node *top, int &val); // Delete stack
void show_Stack(node *top); // Show stack
void main()
{node *top;
 int val;
 int choice;
 char opt = 'Y'; // To continue the do loop in case
 top = NULL; // Initialization of Stack
 clrscr();
 do
 {
 cout << "\n\t\t Main Menu";
 cout << "\n\t\t1. Addition of Stack";
 cout << "\n\t\t2. Deletion from Stack";
 cout << "\n\t\t3. Traverse of Stack";
 cout << "\n\t\t4. Exit from Menu";
 cout << "\n\nEnter Your choice from above ";
 cin >> choice;
 switch (choice)
 {
 case 1:
 do
 {cout << "Enter the value to be added in the stack ";
 cin >> val;
 top = push(top, val);
 cout << "\nDo you want to add more element <Y/N>? ";
 cin >> opt;
 } while (toupper(opt) == 'Y');
 break;

```

```

case 2:
 opt = 'Y'; // Initialize for the second loop
 do
 {top = pop(top, val);
 if (val != -1)
 cout << "Value deleted from Stack is " << val;
 cout << "\nDo you want to delete more element <Y/N>? ";
 cin >> opt;
 } while (toupper(opt) == 'Y');
 break;
case 3:
 show_Stack(top);
 break;
case 4:
 exit(0);
 }
}
while (choice != 4);
}
node *push(node *top, int val)
{
 node *temp;
 temp = new node;
 temp->data = val;
 temp->link = NULL;
 if(top ==NULL)
 top = temp;
 else
 {
 temp->link = top;
 top = temp;
 }
 return(top);
}
node *pop(node *top, int &val)
{
 node *temp;
 clrscr();
 if (top == NULL)
 {
 cout<<"Stack Empty ";
 val = -1;
 }
 else
 {
 temp = top;
 top = top->link;
 val = temp->data;
 temp->link = NULL;
 delete temp;
 }
 return (top);
}
void show_Stack(node *top)
{
 node *temp;
 temp = top;
 clrscr();
 cout<<"The values are \n";
 while (temp != NULL)
 {
 cout <<"\n"<< temp->data;
 temp = temp->link;
 }
}

```

## Converting INFIX to POSTFIX notation:

An arithmetic expression is an expression having variables, constants and operators. The hierarchy of the operators is as follows:

|                                    |                   |
|------------------------------------|-------------------|
| <b>Arithmetic Operators</b>        | <b>Precedence</b> |
| Exponents (^ or ↑)                 | Highest           |
| Multiplication(*) and Division (/) | Middle            |
| Addition (+) and Subtraction (-)   | Lowest            |
| <b>Logical Operators</b>           | <b>Precedence</b> |
| NOT(!)                             | Highest           |
| AND (&&)                           | Middle            |
| OR (   )                           | Lowest            |

Operators at the same level are executed in a left to right order and can be enclosed with parenthesis to override the above rules.

The usual way of writing expressions is by writing operator between two operands. This notation is called infix notation. In postfix notation operator is placed after the two operands.

Stack can be used to convert an infix expression into postfix notation.

### Steps:

1. Scan the given infix expression from left to right and repeat the step 2 to 5 for each element until the stack is empty.
2. If an operand is encountered, place it onto the output (pop).
3. If a left parenthesis is encountered, push it into stack.
4. If an operator is encountered, then:
  - (i) If a symbol of lower or same priority is encountered, pop entries from the stack until an entry of lower priority is reached.
  - (ii) After popping, push the operator onto stack
5. If a right parenthesis is encountered, pop the stack elements until a corresponding left parenthesis is reached.
6. End

**Example:** Convert  $((A+B)*C/D+E^F)/G$  into postfix notation from showing stack status after every step.

### Answer:

| Step | Input | Action      | Stack Status | Output              |
|------|-------|-------------|--------------|---------------------|
| 1    | (     | Push        | (            |                     |
| 2    | (     | Push        | ((           |                     |
| 3    | A     | Print       | ((           | A                   |
| 4    | +     | Push        | ((+          | A                   |
| 5    | B     | Print       | ((+          | AB                  |
| 6    | )     | Pop & Print | (            | AB+                 |
| 7    | *     | Push        | (*           | AB+                 |
| 8    | C     | Print       | (*           | AB+C                |
| 9    | /     | Pop & print | (            | AB+C*               |
|      |       | Push        | (/           | AB+C*               |
| 10   | D     | Print       | (/           | AB+C*D              |
| 11   | +     | Pop & Print | (            | AB+C*D/             |
| 12   | E     | Print       | (            | AB+C*D/E            |
| 13   | ^     | Push        | (^           | AB+C*D/E            |
| 14   | F     | Print       | (^           | AB+C*D/EF           |
| 15   | )     | Pop & Print | Empty        | AB+C*D/EF^          |
| 16   | /     | Push        | /            | AB+C*D/EF^          |
| 17   | G     | Print       | /            | AB+C*D/EF^G         |
| 18   |       | Pop         | Empty        | <b>AB+C*D/EF^G/</b> |

Remark: \* and / has same priority so in step 9, so \* is popped and after that / is pushed.

**Example:** Convert the expression  $(TRUE \ \&\& \ FALSE) \ || \ !(FALSE \ || \ TRUE)$  into postfix notation from showing stack status after every step.

**Answer:**

| Step | Input | Action | Stack Status | Output                        |
|------|-------|--------|--------------|-------------------------------|
| 1    | (     | Push   | (            |                               |
| 2    | TRUE  | Print  | (            | TRUE                          |
| 3    | &&    | Push   | (&&          | TRUE                          |
| 4    | FALSE | Print  | (&&          | TRUE FALSE                    |
| 5    | )     | Pop    | Empty        | TRUE FALSE &&                 |
| 6    |       | Push   |              | TRUE FALSE &&                 |
| 7    | !     | Push   | !            | TRUE FALSE &&                 |
| 8    | (     | Push   | ! (          | TRUE FALSE &&                 |
| 9    | FALSE | Print  | ! (          | TRUE FALSE && FALSE           |
| 10   |       | Push   | ! (          | TRUE FALSE && FALSE           |
| 11   | TRUE  | Print  | ! (          | TRUE FALSE && FALSE TRUE      |
| 12   | )     | Pop    | !            | TRUE FALSE && FALSE TRUE      |
| 13   |       | Pop    |              | TRUE FALSE && FALSE TRUE    ! |
| 14   |       | Pop    | Empty        | TRUE FALSE && FALSE TRUE    ! |

Remark: Here TRUE , FALSE are operands and !, ||, && are operators.

### Evaluation of a POSTFIX expression:

#### **Steps:**

1. Read the expression from left to right, if an operand is encountered, push it into stack.
2. If a binary operator is encountered, pop two operands from stack and if a unary operator is encountered, pop one operand and then evaluate it.
3. Push back the result onto stack.
4. Repeat it till the end of the expression.

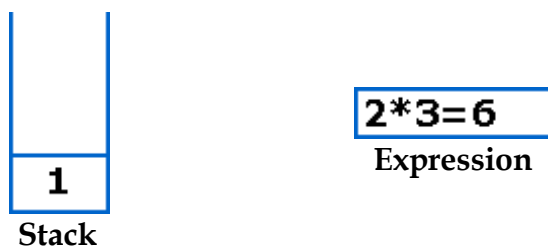
Let us see how the above algorithm will be implemented using an example.

Postfix String : 123\*+4-

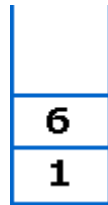
Initially the Stack is empty. Now, the first three characters scanned are 1,2 and 3, which are operands. Thus they will be pushed into the stack in that order.



Next character scanned is "\*", which is an operator. Thus, we pop the top two elements from the stack and perform the "\*" operation with the two operands. The second operand will be the first element that is popped.

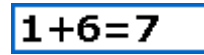


The value of the expression  $(2*3)$  that has been evaluated (6) is pushed into the stack.



**Expression Stack**

Next character scanned is "+", which is an operator. Thus, we pop the top two elements from the stack and perform the "+" operation with the two operands. The second operand will be the first element that is popped.



**Expression Stack**

The value of the expression  $(1+6)$  that has been evaluated (7) is pushed into the stack.



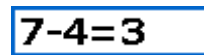
**Expression Stack**

Next character scanned is "4", which is added to the stack.



**Expression Stack**

Next character scanned is "-", which is an operator. Thus, we pop the top two elements from the stack and perform the "-" operation with the two operands. The second operand will be the first element that is popped.



**Expression Stack**

The value of the expression  $(7-4)$  that has been evaluated (3) is pushed into the stack.

Now, since all the characters are scanned, the remaining element in the stack (there will be only one element in the stack) will be returned.

End result :

Postfix String :  $123*+4-$

Result : 3

**Example:** Evaluate the following postfix notation of expression (show status of stack after execution of each operation).

5, 11, -, 6, 8, +, 12, \*, /

**Answer:**

| Step | Input | Action    | Stack Status |
|------|-------|-----------|--------------|
| 1    | 5     | Push (5)  | 5            |
| 2    | 11    | Push (11) | 5, 11        |
| 3    | -     | Pop (11)  |              |
|      |       | Pop (5)   |              |

|   |    |                |              |
|---|----|----------------|--------------|
|   |    | Push (5-11)    | -6           |
| 4 | 6  | Push (6)       | -6, 6        |
| 5 | 8  | Push (8)       | -6, 6, 8     |
| 6 | +  | Pop (8)        | -6, 6        |
|   |    | Pop (6)        | -6           |
|   |    | Push (6+8)     | -6, 14       |
| 7 | 12 | Push (12)      | -6, 14, 12   |
| 8 | *  | Pop (12)       | -6, 14       |
|   |    | Pop (14)       | -6           |
|   |    | Push (14 * 12) | -6, 168      |
| 9 | /  | Pop (168)      | -6           |
|   |    | Pop (-6)       |              |
|   |    | Push (-6/ 168) | <b>-1/28</b> |

**Example:** Evaluate the following postfix notation of expression (show status of stack after execution of each operation).

TURE, FALSE, TRUE, FALSE, NOT, OR, TRUE, OR, OR, AND

**Answer:**

| Step | Input | Action                    | Stack Status             |
|------|-------|---------------------------|--------------------------|
| 1    | TRUE  | Push                      | TRUE                     |
| 2    | FALSE | Push                      | TRUE, FALSE              |
| 3    | TRUE  | Push                      | TRUE, FALSE, TRUE        |
| 4    | FALSE | Push                      | TRUE, FALSE, TRUE, FALSE |
| 5    | NOT   | Pop(False)                | TRUE, FALSE, TRUE        |
|      |       | Push (Not FALSE=TRUE)     | TRUE, FALSE, TRUE, TRUE  |
| 6    | OR    | Pop(TRUE)                 | TRUE, FALSE, TRUE        |
|      |       | Pop (TRUE)                | TRUE, FALSE              |
|      |       | push (TRUE or TRUE=TRUE)  | TRUE, FALSE, TRUE        |
| 7    | TRUE  | Push                      | TRUE, FALSE, TRUE, TRUE  |
| 8    | OR    | pop (TRUE)                | TRUE, FALSE, TRUE        |
|      |       | pop(TRUE)                 | TRUE, FALSE,             |
|      |       | push (TRUE or TRUE=TRUE)  | TRUE, FALSE, TRUE        |
| 9    | OR    | pop (TRUE)                | TRUE, FALSE,             |
|      |       | pop(FALSE)                | TRUE                     |
|      |       | push (TRUE or FALSE=TRUE) | TRUE, TRUE               |
| 10   | AND   | pop (TRUE)                | TRUE,                    |

|  |                           |      |
|--|---------------------------|------|
|  | pop(TRUE)                 |      |
|  | push (TRUE and TRUE=TRUE) | TRUE |

### UNSOLVED PROBLEMS

- Problem 1: Write a function in C++ to insert an element into a dynamically allocated Stack where each node contains a real number as data. Assume the following definition of STACKNODE for the same.
- ```
struct STACKNODE
{
float DATA;
STACKNODE *LINK;
};
```
- Problem 2: Write a function in C++ to delete a node containing Book's Information , from a dynamically allocated Stack of Books implemented with the help of the following structure:
- ```
struct book
{
int bno;
char bname[20];
book * next;
};
```
- Problem 3: Give the Postfix form of the following expression showing stack status:  
 $A*(B + (C + D) * (E * F)/G) * H$
- Problem 4: Evaluate the following postfix notation of expression (Show stack status after execution of each operation):  
3, 9, 4, +, \*, 10, 2, /, -
- Problem 5: Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) :  
50,40,+,18,14,-, 4,\*,+
- Problem 6: Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) :  
45, 7, +, 8, 10, -, \*
- Problem 7: Evaluate the following postfix notation of expression (Show status of stack after execution of each operation) :  
True,False,AND,True,True,NOT,OR,AND
- Problem 8: Write a function in C++ to insert an element in an array stack.



## : Queue

### **QUEUE**

Introduction, array implementation of queue, linked list implementation of queue, circular queue

### **Introduction:**

A queue is a subclass of lists in which insertion and deletion take place at specific ends i.e. REAR and FRONT respectively. It is a FIFO (First In First Out) data structure. By convention, we name the queue insert operation *enqueue* and the remove operation *dequeue*.



### **Array Implementation of Queue:**

Array is a static data structure. So the space is allocated according to the maximum number of elements present at that point.

Insertion of an element in the queue is possible through REAR end. In case the array is full and no new element can be accommodated, it is called QUEUE-FULL. This condition is called OVERFLOW.

Deletion of an element takes place from the FRONT. In case the last element is deleted, the queue becomes empty. If one tries to delete an element from an empty queue, this is called UNDERFLOW.

// Program illustrating basic operations in an array queue

```
#include<iostream.h>
#include<conio.h>
#include <stdlib.h>
class queue
{
 int data[10];
 int front, rear;
public:
 queue()
 {
 front = -1;
 rear = -1;
 }
 void add(); // To add an element into the queue
 void remove(); // To remove an element from the wueue
 void Delete(int ITEM); //To delete all elements which are equal to ITEM;
};
void queue::add()
{
 if (rear == front)
 {
 if (rear == -1)
 front = rear = 0;
 else
 rear = (rear + 1) % 10;
 cout << "Enter data : ";
 cin >> data[rear];
 }
 else
 cout << "Queue full :: Overflow error!!\n";
}
void queue::remove()
{
 if (front != -1)
 {
```

```

 cout << data[front] << " deleted ";
 if (front == rear)
 front = rear - 1;
 else
 front = (front - 1) % 10;
 }
 else
 cout << "Queue empty ! Underflow error!!\n";
}
void main()
{
 clrscr();
 queue Q;
 Q.add();
 Q.remove();
}

```

### **Linked List Implementation of Queue:**

A linked list is a dynamic data structure. So the space requirement is not predetermined. The linked-queue is created after getting a node for the ITEM to be inserted. REAR points to the newly inserted node.

```

//Program illustrating basic operations in a linked queue
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
struct node
{
 char data;
 node *link;
};
node *add_Q(node *rear, char val); // Add queue
node *del_Q(node *front, char &val); // Delete queue
void show_Q(node *front); // Show queue
void main()
{
 node *front, *rear;
 char val;
 int choice;
 char opt = 'Y'; // To continue the do loop in case
 front = rear = NULL; // Initialization of Queue
 clrscr();
 do
 {
 cout << "\n\t\t Main Menu";
 cout << "\n\t1. Addition of Queue";
 cout << "\n\t2. Deletion from Queue";
 cout << "\n\t3. Traverse of Queue";
 cout << "\n\t4. Exit from Menu";
 cout << "\n\nEnter Your choice from above ";
 cin >> choice;
 switch (choice)
 {
 case 1:
 do
 {
 cout << "Enter the value to be added in the queue ";
 cin >> val;
 rear = add_Q(rear, val);
 if (front == NULL)
 front = rear;
 cout << "\nDo you want to add more element <Y/N>? ";
 cin >> opt;
 } while (toupper(opt) == 'Y');

```

```

 break;
 case 2:
 opt = 'Y'; // Initialize for the second loop
 do
 {
 front = del_Q(front, val);
 if (front == NULL)
 rear = front;
 if (val != -1)
 cout << "Value deleted from Queue is " << val;
 cout << "\nDo you want to delete more element <Y/N>? ";
 cin >> opt;
 } while (toupper(opt) == 'Y');
 break;
 case 3:
 show_Q(front);
 break;
 case 4:
 exit(0);
 }
}
while (choice != 4);
}
node *add_Q(node *rear, char val)
{
 node *temp;
 temp = new node;
 temp->data = val;
 temp->link = NULL;
 rear->link = temp;
 rear = temp;
 return (rear);
}
node *del_Q(node *front, char &val)
{
 node *temp;
 clrscr();
 if (front == NULL)
 {
 cout << "Queue Empty ";
 val = -1;
 }
 else
 {
 temp = front;
 front = front->link;
 val = temp->data; temp-
 >link = NULL; delete
 temp;
 }
}

```

```

 return (front);
 }
void show_Q(node *front)
{
 node *temp; temp
 = front;
 clrscr();
 cout << "The Queue values are"; while
 (temp != NULL)
 {
 cout << "\n" << temp->data; temp
 = temp->link;
 }
}

```

### Circular Queue:

When queue is implemented using array a situation arises when overflow occurs whenever though the free cells are available. To overcome this drawback, we have circular queue. In a circular queue the first element of the array is assumed to be immediately next to its last element, i.e. if the last position of the array is having an element, a new element can be inserted next to it, at the first position of the array (if empty).

```

// Program illustrating basic operation of circular queue
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define MAX 100
char queue[MAX][MAX];
int front, rear;
void add_Q(char queue[MAX][MAX], int front, char val[MAX], int &rear); void
del_Q(char queue[MAX][MAX], int &front, int rear);
void show_Q(char queue[MAX][MAX], int front, int rear);
void main()
{
 int choice; char
 data[MAX];
 char opt = 'Y'; // To continue the do loop in case
 rear = -1; // Initialization of Queue
 front = -1;
 clrscr(); do
 {
 cout << "\n\t\t Main Menu";
 cout << "\n\t1. Addition of Queue";
 cout << "\n\t2. Deletion from Queue";
 cout << "\n\t3. Traverse of Queue";
 cout << "\n\t4. Exit from Menu";
 cout << "\n\nEnter Your choice from above ";
 cin >> choice;
 switch (choice)
 {
 case 1:
 do
 {

```

```

cout << "Enter the character string : "; gets(data);
add_Q(queue, front, data, rear);
cout << "Do you want to add more element <Y/N> ? "; cin >> opt;
} while (toupper(opt) == 'Y'); break;
 case 2:
 opt = 'Y';
 do
 {
 del_Q(queue, front, rear);
 cout << "\nDo you want to delete more element <Y/N> ? ";
 cin >> opt;
 } while (toupper(opt) == 'Y');
 break;
 case 3:
 show_Q(queue, front, rear);
 break;
 case 4:
 exit(0);
}
}

while (choice != 4);
void add_Q(char queue[MAX][MAX], int front, char val[MAX], int &rear)
{if ((rear + 1) % MAX == front)
{
 cout << "Queue Full ";
}
else
{
 rear = (rear + 1) % MAX;
 strcpy(queue[rear], val);
}
}
void del_Q(char queue[MAX][MAX], int &front, int rear)
{
char value[MAX];
if (front == rear)
{
 cout << "Queue Empty ";
}
else
{
 front = (front + 1) % MAX;
 strcpy(value, queue[front]);
 cout << "The value deleted from the queue is "<<value;
}
}
void show_Q(char queue[MAX][MAX], int front, int rear)
{clrscr();
cout << "The values are ";
do
{
 front = (front + 1) % MAX;
 cout << "\n" << queue[front];
}while(front != rear);
}

```

## UNSOLVED PROBLEMS

- Problem 1: Write a insert function in c++ in a dynamically allocated Queue containing Phonno and names of customer.
- Problem 2: Write a function in C++ to delete an element from a dynamically allocated circular Queue where each node contains a real number as data.
- Problem 3: Write a function to perform a Push operation in a dynamically allocated stack considering the following description struct node { int data; struct node \* next; }; 2.
- Problem 4: Write a function in C++ to perform Insert operation in a dynamically allocated Queue containing names of students.

## Unit 3: Database And SQL

### : Database Concepts

#### **DATABASE CONCEPTS**

Terminology, hierarchical model, network model, relational model

#### Terminology:

- A database consists of a number of **tables**. Each table comprises of rows(records) and columns(attributes). Each record contains values for the corresponding attributes. The values of the attributes for a record are interrelated. For example, different cars have different values for the same specifications (length, color, engine capacity, etc.).
- In the database oriented approach, we store the common data in one table and access it from the required tables. Thus the **redundancy of data decreases**.
- The database oriented approach **supports multiple views** of the same data. For example, a clerk may only be able to see his details, whereas the manager can view the details of all the clerks working under him.
- Multiple views of the same database may exist for different users. This is defined in the view level of abstraction.
- The **logical level of abstraction** defines the type of data that is stored in the database and the relationship between them.
- The design of the database is known as the **database schema**.
- The **instance of the database** is the data contained by it at that particular moment.
- The **Database Administrator** has the total control of the database and is responsible for the setting up and maintaining the database.
- A **Data Model** is the methodology used by a particular DBMS to organize and access the data
- **Hierarchical, Network and Relational Model** are the three popular data models. However, the relational model is more widely used.

#### Hierarchical Model:

- The hierarchical model was developed by **IBM** in 1968.
- The data is organize in a **tree structure** where the nodes represent the records and the branches of the tree represent the fields.
- Since the data is organized in a tree structure, the parent node has the **links** to its child nodes.
- If we want to search a record, we have to traverse the tree from the root through all its parent nodes to reach the specific record. Thus, **searching for a record is very time consuming**.
- The **hashing function** is used to locate the root.
- **SYSTEM2000** is an example of hierarchical database

### Network Model:

- **Record relationship** in the network model is implemented by using **pointers**.
- Record relationship implementation is very complex since pointers are used. It supports many-to-many relationships and **simplified searching of records** since a record has many access paths.
- **DBTG Codasyl** was the first network database.

### Relational Model:

- The Relational Model, organizes data in the form of independent **tables** (consisting of rows and columns) that are related to each other.
- A table consists of a number of **rows (records/tuples) and columns (attributes)**. Each record contains values for the attributes.
- The **degree of the table** denotes the number of columns.
- A domain in the relational model is said to be **atomic** if it consists of indivisible units. For example, name is not atomic since it can be divided into first name and last name.
- E. F. Codd laid down 12 rules (known as **Codd's 12 rules**) that outline the minimum functionality of a RDBMS. A RDBMS must comply with at least 6 of the rules.
- A **Super Key** is a set of attributes that collectively identify an entity in an entity set. For example, the bank account number is a super key in the bank accounts table.
- A **Candidate Key** (also known as **Primary Key**) is the smallest subset of the super key for which there does not exist a proper subset that is a super key.
- Out of the multiple candidate keys, only one is selected to be the primary key and the remaining are **alternate keys**.
- A **foreign key** is the primary key of a table that is placed into a related table to represent one-to-many relationship among these tables.

## UNSOLVED PROBLEMS

- Problem 1:           What is the main function of DBA.
- Problem 2:           What is foreign Key? What is its purpose
- Problem 3:           Define the terms Tuple and Attribute
- Problem 4:           What do you understand by the terms Cardinality and Degree of the table?



## : Structure Query Language

### **SQL**

What is SQL, types of SQL commands –DDL, DML, DCL, TCL, Concept of SQL, Basic structure of SQL query, Type of SQL commands, Constraints,

### What is SQL?

- When a user wants to get some information from a database file, he can issue a query.
- A query is a user-request to retrieve data or information with a certain condition.
- SQL is a query language that allows user to specify the conditions. (instead of algorithms)

### Types of SQL commands:

**Data Definition Language commands (DDL Command):** All the commands used to create, modify or delete physical structure of an object like Table.

For eg. Create, Alter, drop

**Data Manipulation Language command (DML Command):** All the commands used to modify contents of a table are comes under this category.

For eg. : Insert, delete, update commands

**TCL Command:** These commands are used to control Transaction of DML commands.

For eg. Commit, rollback

### Concept of SQL:

- The user specifies a certain condition.
- The program will go through all the records in the database file and select those records that satisfy the condition.(searching).
- Statistical information of the data.
- The result of the query will then be stored in form of a table.

### Basic structure of an SQL query:

|                          |                                                       |
|--------------------------|-------------------------------------------------------|
| <b>General Structure</b> | SELECT, ALL / DISTINCT, *,AS, FROM, WHERE             |
| <b>Comparison</b>        | IN, BETWEEN, LIKE "% _"                               |
| <b>Grouping</b>          | GROUP BY, HAVING, COUNT(), SUM(), AVG(), MAX(), MIN() |
| <b>Display Order</b>     | ORDER BY, ASC / DESC                                  |
| <b>Logical Operators</b> | AND, OR, NOT                                          |

### TYPES OF SQL STATEMENTS:

- a) DDL (Data Definition Language):- Create ,Alter,Drop.
- b) DML (Data Manipulation Language):- Select,Delete,Insert,Update.
- c) DCL (Data Control Language):- Grant,Revoke.
- d) TCL (Transaction Control Language):- COMMIT,ROLLBACK,SAVEPOINT.

### CONSTRAINT:

Constraint is a condition applicable on a field or group of fields.

Two types of constraint:

**Column Constraint** :- apply only to individual column

**Table Constraint** :- apply to groups of columns

Different constraint

Unique Constraint Primary Key constraint

Default constraint Check constraint

Applying Constraint

Example:-

Create a student table with filed student id, student name, father" s name, age, class, adress.

```
CREATE TABLE student
(sid char(4) PRIMARY KEY,
sname char(20) NOT NULL,
fname char(20),
age number(2) CHECK (age<20),
class char(5) NOT NULL ,
address char(50));
```

**SELECT COMMAND**

Select command is a query that is given to produce certain specified information from the database table.

Select Statement is used as

```
SELECT <column name>,[<column name>,.....]
FROM <table name>;
```

**Example:** Write a query to display the name and salary of the employee in emp table.

```
SELECT ename, sal
FROM emp;
```

Variations of select Command:

Selecting specific Rows.....WHERE clause

**Syntax:**

```
SELECT <column-name>[,<column-name>,.....]
FROM <table name>
WHERE <condition>;
```

**Example :** Display the employee code, their name and their salary who are Manager.

```
SELECT empno,ename,sal
FROM emp
WHERE job=" MANAGER" ;
```

Searching for NULL (IS NULL Command):

The null value in a column can be searched for in a table using **IS NULL** in the WHERE Clause

Syntax:

```
SELECT <column-name>[,<column-name>,.....]
FROM <table-name>
WHERE <column-name> IS NULL;
```

**Example** Display the employee code, name and their job whose Dept.No. is Null.

```
SELECT empno,empname,job
FROM emp
WHERE DeptNo IS NULL;
```

IS NOT NULL Command:

**Example:** Display the name and job of those employees whose dept No is not Null  
SELECT ename,job FROM emp

WHERE deptno IS NOT NULL;

#### Logical Operators

The logical operators **OR**, **AND**, **NOT** are used to connect search conditions in the WHERE clause. The uses of logical operators are understood by these following **examples**

Display the name of manager whose salary is more than 5000

```
SELECT ename
FROM emp
WHERE job=" MANAGER" and sal>5000;
```

Write a query on the customers table whose output will exclude all customers with rating<=100, unless they are located in Shimla.

```
SELECT *
FROM customers
WHERE rating>100 OR city=" Shimla" ;
```

#### Sorting Result- ORDER BY Clause:

The resulting column can be sorted in ascending and descending order using the ORDER BY Clause.

#### Syntax :

```
SELECT <column-name>[,<column-name>.....]
FROM <table name>
WHERE <condition>
ORDER BY <column-name>
```

#### **Example:**

Display the list of employee in the descending order of employee code, who is manager

```
SELECT * FROM emp
WHERE job=" MANAGER"
ORDER BY ecode;
```

#### The INSERT Command:

The tuples are added to relation using INSERT command of SQL.

#### Syntax:

```
INSERT INTO <table-name>[<column list>]
VALUES (<value>,<value>,<value>,.....);
```

#### **Example :**

Enter a new record in student table

```
INSERT INTO student (sid,sname,fname,age,class,address);
VALUES(101," Mohan" ," Pawan" ,15," 8" ," Jaipur");
```

sid sname fname age class address

101 Mohan Pawan 15 8 Jaipur

#### The DELETE Command:

The delete command removes the tuples from the tables. This command removes the entire row from the table and not the individual field. So no field argument is needed.

#### Syntax

```
DELETE FROM <table-name>
WHERE <condition>;
```

#### **Example**

Delete all the records of employee whose salary is less than 3000

```
DELETE FROM emp
WHERE sal<3000;
```

To delete all the record from the table:  
DELET FROM<table-name>;

The UPDATE Command:

The UPDATE command is used to changes some values in existing rows. The UPDATE command specifies the rows to be changed using the WHERE clause, and new data using the SET keyword.

**Example:**

Update the salary of employee to 5000 whose employee code is 1011.

```
UPDATE emp
SET sal=5000
WHERE empno=1011;
```

The ALTER TABLE Command:

The ALTER command is used to change the definition of existing table.

a)It can be used to add columns to a table.

**Syntax** (to add a column to a table):

```
ALTER TABLE <table-name> ADD <column-name>
<data type> <size>;
```

b)To modify existing columns of a table:

Syntax:

```
ALTER TABLE <table-name>
MODIFY (Columnname newdatatype (news�));
```

Example:

To modify column job of table emp to have new width of 30 character

```
ALTER TABLE emp
MODIFY (job char(30));
```

The DROP Command

The DROP command is used to drop the table from the database. For dropping a table all the tuples should be deleted first i.e the table should be empty.

Syntax:

```
DROP TABLE <table-name>
```

Example :

Drop the student table from the database

```
DROP TABLE student;
```

**Some Example:**

Write a query on the customers table whose output will exclude all customers with a rating <=100, unless they are located in Shimla.

Ans. SELECT \* FROM customers WHERE rating >100 OR city ="Shimla" ;

Write a query that selects all orders except those zeros or NULLs in the amount field.

Ans. SELECT \* FROM Orders WHERE amt < >0 AND (amt IS NOT NULL) ;

Write a query that lists customers in descending order of rating.

Output the rating field first, followed by the customer's name and number.

Ans. SELECT rating, cust-name, cust-num FROM customers ORDER BY rating DESC ;

Write a command that puts the following values, in their given order, into the salesman table: cust-name-Manisha, city-Manali, comm.- NULL, cust-num-1901.

Ans. INSERT INTO salesman (city, cust-name, comm.,cust-num)
VALUES(,Manisha",NULL,1901) ;

## UNSOLVED PROBLEMS

Problem 1: What are DDL and DML?

Problem 2: What is the difference between Where and Having

Clause ? Problem 3: Write the SQL query commands based on following table

**Table : Book**

| Book_id | Book name       | Author_name     | Publisher    | Price | Type    | Quantity |
|---------|-----------------|-----------------|--------------|-------|---------|----------|
| C0001   | Fast Cook       | Lata Kapoor     | EPB          | 355   | Cookery | 5        |
| F0001   | The Tears       | William Hopkins | First Publi. | 650   | Fiction | 20       |
| T0001   | My First c++    | Brain & Brooke  | FPB          | 350   | Text    | 10       |
| T0002   | C++ Brain works | A.W. Rossaine   | TDH          | 350   | Text    | 15       |
| F0002   | Thunderbolts    | Anna Roberts    | First Publ.  | 750   | Fiction | 50       |

**Table : issued**

| Book_Id      | Quantity Issued |
|--------------|-----------------|
| <b>T0001</b> | <b>4</b>        |
| <b>C0001</b> | <b>5</b>        |
| <b>F0001</b> | <b>2</b>        |

Write SQL query for (a) to (f)

- (a) To show book name, Author name and price of books of First Pub. Publisher
- (b) To list the names from books of text type
- (c) To Display the names and price from books in ascending order of their prices.
- (d) To increase the price of all books of EPB publishers by 50.
- (e) To display the Book\_Id, Book\_name and quantity issued for all books which have been issued
- (f) To insert a new row in the table issued having the following data. „F0003“, 1
- (g) Give the output of the following
  - i. Select Count(\*) from Books
  - ii. Select Max(Price) from books where quantity >=15
  - iii. Select book\_name, author\_name from books where publishers="first publ."
  - iv. Select count(distinct publishers) from books where Price >=400

Problem 4: What are group Functions?

Problem 5: What do you understand by constraints?

## Unit 4: Boolean Algebra

### BOOLEAN ALGEBRA

Introduction, Binary valued Quantities, Logical Operator, Truth Table, Logical Operators, logic gates, Principle of duality, Basic theorem of Boolean Algebra, minterm, maxterm, Canonical form (SOP and POS), Karnaugh Map, More about gates, Combination of gates,

#### Introduction:

In 1938 Claude E. Shannon wrote a paper titled „A Symbolic Analysis of Relay Switching Circuits” . In this paper he applied Boolean algebra to solve relay logic problems. As logic problems are binary decisions and Boolean algebra effectively deals with these binary values. Thus it is called „Switching Algebra” .

#### Binary Valued Quantities:

The decision which results into either YES (TRUE) or NO( False) is called a Binary Decision. Example:- I have to go or Not? In this we have two decisions either for True or for False.

#### Logical Operator:

In Algebraic function we use +, -, \*, / operator but in case of Logical Function or Compound statement we use AND, OR & NOT operator.

Example: He prefers Computer Science NOT IP.

#### Truth Table:

The table which is used for calculation of results in true & False for all combination of Input variable.

Example:-

| X | Y | R |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

1(One) represents TRUE values and 0(Zero) represents FALSE value.

If result of any logical statement or expression is always TRUE or 1, it is called **Tautology** and if the result is always FALSE or 0 it is called **Fallacy**.

#### Logical Operator:

There are three Basic Logical Operator:

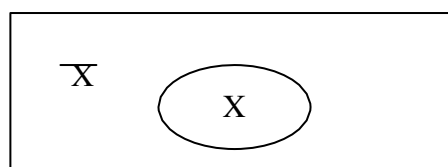
1. NOT
2. OR
3. AND

NOT Operator

This operator operates on single variable and operation performed by not operator is called Complementation i.e. for 0 to 1 or 1 to 0.

Truth Table:-

| X | R |
|---|---|
| 0 | 1 |
| 1 | 0 |





Example: - The best Example of Not gate is Inverter which is used in our Home.

### OR Operator

The OR operator gives the True result if all or anyone of the input signal is true, it is used as a logical addition and denoted by +.

$$0+0=0$$

$$0+1=1$$

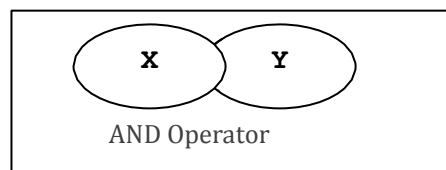
$$1+0=1$$

$$1+1=1$$

Truth Table:-

| X | Y | R |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Venn Diagram for X+Y is



AND operator gives the True result if all inputs are one or true otherwise false, this is known as logical multiplication denoted by DOT (.) Operator.

$$0.0=0$$

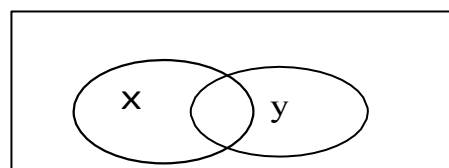
$$0.1=0$$

$$1.0=0$$

$$1.1=1$$

Truth Table :-

| X | Y | R |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



Overlapped Portion shows X.Y

Evaluation of Boolean Expressions Using Truth Table

### Logic Gate:

A gate is simply an electronic circuit which operates on one or more signals to produce an output signal.

### NOT gate (inverter)

The output Q is true when the input A is NOT true, the output is the inverse of the input:

$$Q = \text{NOT } A$$

A NOT gate can only have one input. A NOT gate is also called an inverter.



| Input A | Output Q |
|---------|----------|
| 0       | 1        |
| 1       | 0        |

Traditional symbol

Truth Table

### AND gate

The output Q is true if input A AND input B are both true:  $Q = A \text{ AND } B$   
 An AND gate can have two or more inputs, its output is true if all inputs are true.



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 0        |
| 0       | 1       | 0        |
| 1       | 0       | 0        |
| 1       | 1       | 1        |

Traditional symbol

Truth Table

### OR gate

The output Q is true if input A OR input B is true (or both of them are true):  $Q = A \text{ OR } B$   
 An OR gate can have two or more inputs, its output is true if at least one input is true.



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 0        |
| 0       | 1       | 1        |
| 1       | 0       | 1        |
| 1       | 1       | 1        |

Traditional symbol

Truth Table

### Principle of Duality:

This principle tells us starting with the one relation or gate other can be derived by

1. Changing each OR sign (+) to an AND sign (.)
2. Changing each AND sign to an OR sign (.)
3. Replacing each 0 by 1 and each 1 by 0.

The derived relation using duality principle is called dual of original expression.

Example:-  $0+0=0$  by applying 0 to 1 & + to . it becomes  $1.1=1$

### Basic Theorem of Boolean Algebra

In working with logic relations in digital form, we need a set of rules for symbolic manipulation which will enable us to simplify complex expressions and solve for unknowns. Originally, Boolean algebra which was formulated by **George Boole**, an English mathematician (1815-1864) described propositions whose outcome would be either *true* or *false*. In computer work it is used in addition to describe circuits whose state can be either *1 (true)* or *0 (false)*. Using the relations defined in the AND, OR and NOT operation, a number of postulates are stated in

- P1 :  $X = 0$  or  $X = 1$
- P2 :  $0.0 = 0$
- P3 :  $1 + 1 = 1$

- **P4 :  $0 + 0 = 0$**
- **P5 :  $1 \cdot 1 = 1$**
- **P6 :  $1 \cdot 0 = 0 \cdot 1 = 0$**
- **P7 :  $1 + 0 = 0 + 1 = 1$**
  
- **T1 : Commutative Law**  
 $A + B = B + A$   
 $A \cdot B = B \cdot A$
- **T2 : Associative Law**  
 $(A + B) + C = A + (B + C)$   
 $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- **T3 : Distributive Law**  
 $A \cdot (B + C) = A \cdot B + A \cdot C$   
 $A + (B \cdot C) = (A + B) \cdot (A + C)$
- **T4 : Identity Law**  
 $A + A = A$   
 $A \cdot A = A$
- **T5 : Negation Law**
  
- **T6 : Redundance Law**  
 $A + A \cdot B = A$   
 $A \cdot (A + B) = A$
- **T7 :**  
 $0 + A = A$   
 $1 \cdot A = A$   
 $1 + A = 1$   
 $0 \cdot A = 0$
- **T8 :**
  
- **T9 :**
  
- **T10 : De Morgan's Theorem**

### minterm:

Minterm is a Product of all the literals within the logic System.

Step involved in minterm expansion of Expression

1. First convert the given expression in sum of product form.
2. In each term if any variable is missing (e.g. in the following example Y is missing in first term and X is missing in second term), multiply that term with (missing term + complement (missing term)) factor e.g. if Y is missing multiply with  $Y + Y'$  )
3. Expand the expression .
4. Remove all duplicate terms and we will have minterm form of an expression.

Example: Convert  $X + Y$

$$\begin{aligned}
 X + Y &= X \cdot 1 + Y \cdot 1 \\
 &= X \cdot (Y + Y') + Y \cdot (X + X') \\
 &= XY + XY' + XY + X'Y \\
 &= XY + XY' + X'Y
 \end{aligned}$$

Other procedure for expansion could be

1. Write down all the terms
2. Put  $X''$  s where letters much be inserted to convert the term to a product term
3. Use all combination of  $X''$  s in each term to generate minterms
4. Drop out duplicate terms

### Shorthand Minterm notation:

Since all the letters must appear in every product, a shorthand notation has been developed that saves actually writing down the letters themselves. To form this notation, following steps are to be followed:

1. First of all, Copy original terms
2. Substitute 0'' s for barred letters and 1'' s for nonbarred letters
3. Express the decimal equivalent of binary word as a subscript of m.

Rule1. Find Binary equivalent of decimal subscript e.g.,for  $m_6$  subscript is 6, binary equivalent of 6 is 110.

Rule 2. For every 1'' s write the variable as it is and for 0'' s write variable'' s complemented form i.e., for 110 t is XYZ. XYZ is the required minterm for  $m_6$ .

### maxterm:

A Maxterm is a sum of all the literals (with or without the bar) within the logic system. Boolean Expression composed entirely either of Minterms or Maxterms is referred to as Canonical Expression.

### Canonical Form:

Canonical expression can be represented is derived from

- (i) Sum-of-Products(SOP) form
- (ii) Product-of-sums(POS) form

#### Sum of Product (SOP)

1. Various possible input values
2. The desired output values for each of the input combinations

| X | Y | R        |
|---|---|----------|
| 0 | 0 | $X''Y''$ |
| 0 | 1 | $X''Y$   |
| 1 | 0 | $XY''$   |
| 1 | 1 | $XY$     |

#### Product of Sum (POS)

When a Boolean expression is represented purely as product of Maxterms, it is said to be in Canonical Product-of-Sum form of expression.

| X | Y | Z | Maxterm       |
|---|---|---|---------------|
| 0 | 0 | 0 | $X+Y+Z$       |
| 0 | 0 | 1 | $X+Y+Z''$     |
| 0 | 1 | 0 | $X+Y''+Z$     |
| 0 | 1 | 1 | $X+Y''+Z''$   |
| 1 | 0 | 0 | $X''+Y+Z$     |
| 1 | 0 | 1 | $X''+Y+Z''$   |
| 1 | 1 | 0 | $X''+Y''+Z$   |
| 1 | 1 | 1 | $X''+Y''+Z''$ |

**Karnaugh Maps:**

Karnaugh map or K Map is a graphical display of the fundamental product in a truth table.

**Example: Reduce the following Boolean expression using K-Map:**

$$F(P,Q,R,S)=\Sigma(0,3,5,6,7,11,12,15)$$

Soln:

|        | R''S''  | R''S   | RS      | RS''   |
|--------|---------|--------|---------|--------|
| P''Q'' | 1<br>0  |        | 1<br>3  |        |
| P''Q   |         | 1<br>5 | 1<br>7  | 1<br>6 |
| PQ     | 1<br>12 |        | 1<br>15 |        |
| PQ''   |         |        | 1<br>11 |        |
|        | 8       | 9      | 10      |        |

This is 1 quad, 2pairs & 2 lock

Quad(m3+m7+m15+m11) reduces to RS

Pair(m5+m7) reduces to P''QS

Pair (m7+m6) reduces to P''QR

Block m0=P''Q''R''S''

M12=PQR''S''

hence the final expressions is  $F=RS + P''QS + P''QR + PQR''S'' + P''Q''R''S''$

**Example:** Reduce the following Boolean expression using K-Map:

$$F(A,B,C,D)=\prod(0,1,3,5,6,7,10,14,15)$$

Soln:

|   |   |   |   |
|---|---|---|---|
| 0 | 0 | 0 |   |
|   | 0 | 0 | 0 |
|   |   | 0 | 0 |
|   |   |   | 0 |

Reduced expressions are as follows:

For pair 1, (A+B+C)

For pair 2, (A''+C''+D)

For Quad 1, (A+D'')

For Quad 2, (B''+C'')

Hence final POS expression will be

$$Y(A,B,C,D)= (A+B+C) (A+\overline{C}+\overline{D}) (A+\overline{D}) (\overline{B}+\overline{C})$$

## More about Gate Gates:

NAND gate (NAND = Not AND)

This is an AND gate with the output inverted, as shown by the 'o' on the output. The output is true if input A AND input B are NOT both true:  $Q = \text{NOT} (A \text{ AND } B)$   
A NAND gate can have two or more inputs, its output is true if NOT all inputs are true.



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 1        |
| 0       | 1       | 1        |
| 1       | 0       | 1        |
| 1       | 1       | 0        |

Traditional symbol

Truth Table

NOR gate (NOR = Not OR)

This is an OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if NOT inputs A OR B are true:  $Q = \text{NOT} (A \text{ OR } B)$   
A NOR gate can have two or more inputs, its output is true if no inputs are true.



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 1        |
| 0       | 1       | 0        |
| 1       | 0       | 0        |
| 1       | 1       | 0        |

Traditional symbol

Truth Table

EX-OR (EXclusive-OR) gate

The output Q is true if either input A is true OR input B is true, **but not when both of them are true**:  $Q = (A \text{ AND NOT } B) \text{ OR } (B \text{ AND NOT } A)$

This is like an OR gate but excluding both inputs being true.

The output is true if inputs A and B are **DIFFERENT**.

EX-OR gates can only have 2 inputs.



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 0        |
| 0       | 1       | 1        |
| 1       | 0       | 1        |
| 1       | 1       | 0        |

Traditional symbol

Truth Table

EX-NOR (EXclusive-NOR) gate

This is an EX-OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if inputs A and B are the **SAME** (both true or both false):

$$Q = (A \text{ AND } B) \text{ OR } (\text{NOT } A \text{ AND NOT } B)$$

EX-NOR gates can only have 2 inputs.



Traditional symbol

| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 1        |
| 0       | 1       | 0        |
| 1       | 0       | 0        |
| 1       | 1       | 1        |

Truth Table

### Summary truth tables

The summary truth tables below show the output states for all types of 2-input and 3-input gates.

| Inputs |   | Output of each gate |      |    |     |       |        |
|--------|---|---------------------|------|----|-----|-------|--------|
| A      | B | AND                 | NAND | OR | NOR | EX-OR | EX-NOR |
| 0      | 0 | 0                   | 1    | 0  | 1   | 0     | 1      |
| 0      | 1 | 0                   | 1    | 1  | 0   | 1     | 0      |
| 1      | 0 | 0                   | 1    | 1  | 0   | 1     | 0      |
| 1      | 1 | 1                   | 0    | 1  | 0   | 0     | 1      |

Note that EX-OR and EX-NOR gates can only have 2 inputs.

| Inputs |   |   | Output of each gate |      |    |     |
|--------|---|---|---------------------|------|----|-----|
| A      | B | C | AND                 | NAND | OR | NOR |
| 0      | 0 | 0 | 0                   | 1    | 0  | 1   |
| 0      | 0 | 1 | 0                   | 1    | 1  | 0   |
| 0      | 1 | 0 | 0                   | 1    | 1  | 0   |
| 0      | 1 | 1 | 0                   | 1    | 1  | 0   |
| 1      | 0 | 0 | 0                   | 1    | 1  | 0   |
| 1      | 0 | 1 | 0                   | 1    | 1  | 0   |
| 1      | 1 | 0 | 0                   | 1    | 1  | 0   |
| 1      | 1 | 1 | 1                   | 0    | 1  | 0   |

### Combinations of logic gates:

Logic gates can be combined to produce more complex functions. They can also be combined to substitute one type of gate for another.

For example to produce an output Q which is true only when input A is true and input B is false, as shown in the truth table on the right, we can combine a NOT gate and an AND gate like this:



| Input A | Input B | Output Q |
|---------|---------|----------|
| 0       | 0       | 0        |
| 0       | 1       | 0        |
| 1       | 0       | 1        |
| 1       | 1       | 0        |

$$Q = A \text{ AND NOT } B$$

### Working out the function of a combination of gates

Truth tables can be used to work out the function of a combination of gates.

For example the truth table on the right show the intermediate outputs D and E as well as the final output Q for the system shown below.



| Inputs |   |   | Outputs |   |   |
|--------|---|---|---------|---|---|
| A      | B | C | D       | E | Q |
| 0      | 0 | 0 | 1       | 0 | 1 |
| 0      | 0 | 1 | 1       | 0 | 1 |
| 0      | 1 | 0 | 0       | 0 | 0 |
| 0      | 1 | 1 | 0       | 1 | 1 |
| 1      | 0 | 0 | 0       | 0 | 0 |
| 1      | 0 | 1 | 0       | 0 | 0 |
| 1      | 1 | 0 | 0       | 0 | 0 |
| 1      | 1 | 1 | 0       | 1 | 1 |

$$D = \text{NOT } (A \text{ OR } B)$$

$$E = B \text{ AND } C$$

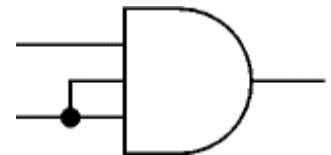
$$Q = D \text{ OR } E = (\text{NOT } (A \text{ OR } B)) \text{ OR } (B \text{ AND } C)$$

### Substituting one type of gate for another

Logic gates are available on ICs which usually contain several gates of the same type, for example four 2-input NAND gates or three 3-input NAND gates. This can be wasteful if only a few gates are required unless they are all the same type. To avoid using too many ICs you can reduce the number of gate inputs or substitute one type of gate for another.

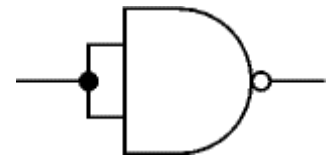
#### Reducing the number of inputs

The number of inputs to a gate can be reduced by connecting two (or more) inputs together. The diagram shows a 3-input AND gate operating as a 2-input AND gate.



#### Making a NOT gate from a NAND or NOR gate

Reducing a NAND or NOR gate to just one input creates a NOT gate. The diagram shows this for a 2-input NAND gate.



### Any gate can be built from NAND or NOR gates

As well as making a NOT gate, NAND or NOR gates can be combined to create any type of gate! This enables a circuit to be built from just one type of gate, either NAND or NOR. For example an AND gate is a NAND gate then a NOT gate (to undo the inverting function). Note that AND and OR gates cannot be used to create other gates because they lack the inverting (NOT) function.

**To change the type of gate**, such as changing OR to AND, you must do three things:

- Invert (NOT) each input.
- Change the gate type (OR to AND, or AND to OR)
- Invert (NOT) the output.

For example an OR gate can be built from NOTed inputs fed into a NAND (AND + NOT) gate.

### NAND gate equivalents

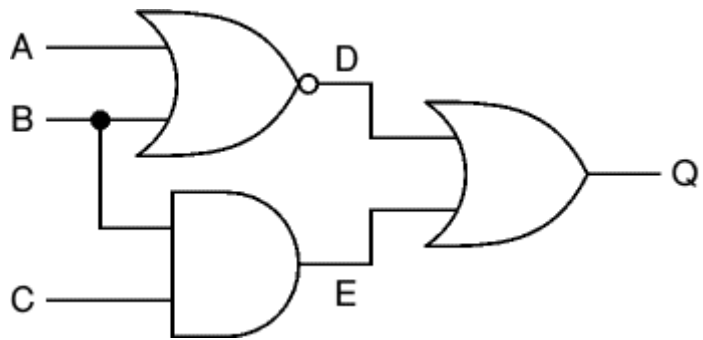
The table below shows the NAND gate equivalents of NOT, AND, OR and NOR gates:



Substituting gates in an example logic system

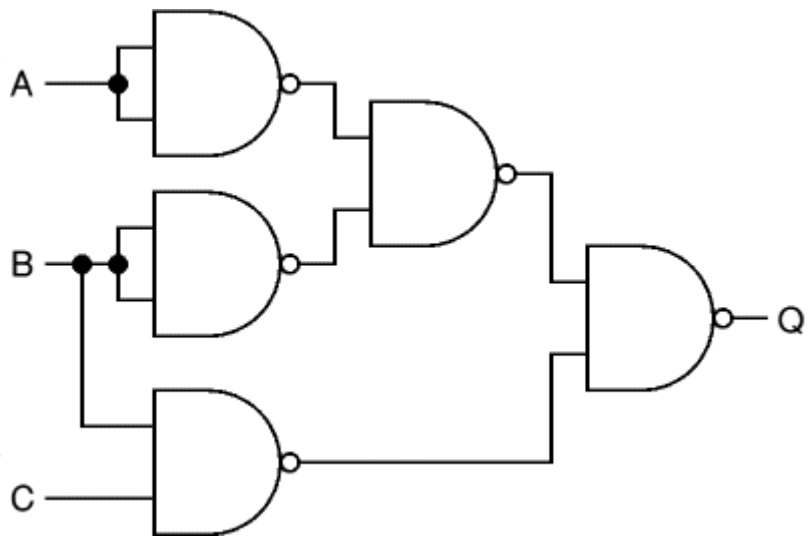
The original system has 3 different gates: NOR, AND and OR. This requires three ICs (one for each type of gate).

To re-design this system using NAND gates only begin by replacing each gate with its NAND gate equivalent, as shown in the diagram below.



Then simplify the system by deleting adjacent pairs of NOT gates (marked X above). This can be done because the second NOT gate cancels the action of the first.

The final system is shown on the right. It has five NAND gates and requires two ICs (with four gates on each IC). This is better than the original system which required three ICs (one for each type of gate).

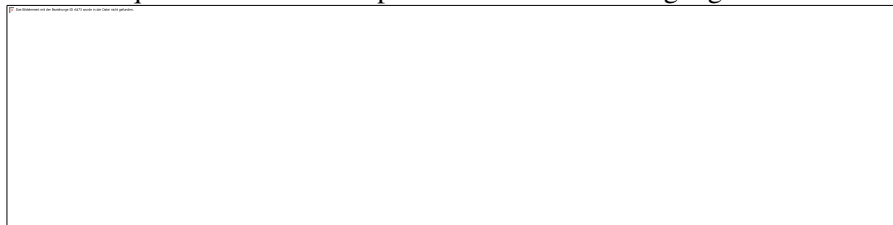


Substituting NAND (or NOR) gates does not always increase the number of gates, but when it does (as in this example) the increase is usually only one or two gates. The real benefit is reducing the number of ICs required by using just one type of gate.

### UNSOLVED PROBLEMS

Problem 1: State and verify absorption law using truth table.

Problem 2: Write the equivalent Boolean Expression for the following logic circuit:



Problem 3: Write the POS form of a Boolean function G, which is represented in a truth table as follows

Problem 4: Reduce the following Boolean expression using K-map:

$$H(U,V,W,Z) = \Sigma(0,1,4,5,6,7,11,12,13,14,15)$$

Problem 5: State and Verify Absorption law in Boolean Algebra.

Problem 6: Draw a logical circuit diagram for the following Boolean Expression:  $A.(B+C'')$

Problem 7: Convert the following Boolean expression into its equivalent Canonical Product of sum form (POS):  $A.B''C + A'' .B.C + A'' .B.C''$ .

Problem 8) Reduce the following Boolean expression using K-map:

$$F(A,B,C,D) = \Sigma(0,1,2,4,5,8,9,10,11)$$

Problem 9) Reduce the following Boolean Expression using K-map.

$$F(A,B,C,D) = \pi(0,2,3,4,6,7,8,10,12)$$

### Unit 5: Communication & Open Source Concepts

#### COMMUNICATION & OPEN SOURCE CONCEPTS

Evolution of networking, switching techniques, data communication terminologies, transmission media, network devices, network topology and types, types of network, network protocols, network security concepts, web services, open source concepts

#### Evolution of Networking:

1. **ARPANET:** In 1969, The US govt. formed an agency named ARPANET (Advanced Research Projects Agency NETwork) to connect computers at various universities and defense agencies. The main objective of ARPANET was to develop a network that could continue to function efficiently even in the event of a nuclear attack.
2. **Internet (INTERconnection NETwork):** The Internet is a worldwide network of computer networks. It is not owned by anybody.
3. **Interspace:** InterSpace is a client/server software program that allows multiple users to communicate online with real - time audio, video and text chat in dynamic 3D environments.

#### Switching Techniques:

1. **Circuit Switching:** In the Circuit Switching technique, first, the complete end-to-end transmission path between the source and the destination computers is established and then the message is transmitted through the path. The main advantage of this technique is guaranteed delivery of the message. Mostly used for voice communication.
2. **Message Switching:** In the Message switching technique, no physical path is established between sender and receiver in advance. This technique follows the store and forward mechanism.
3. **Packet Switching:** In this switching technique fixed size of packet can be transmitted across the network.

Comparison between the Various Switching Techniques:

| Criteria                        | Circuit Switching | Message Switching | Packet Switching |
|---------------------------------|-------------------|-------------------|------------------|
| Path established in advance     | Yes               | No                | No               |
| Store and forward technique     | No                | Yes               | Yes              |
| Message follows multiple routes | No                | Yes               | Yes              |

#### Data Communication terminologies:

1. **Concept of Channel:** - A data channel is the medium used to carry information or data from one point to another.
2. **Baud:** It is the unit to measure the data transmission speed. It is equivalent to bps (bits per second).
3. **Bandwidth:** - The **maximum** volume of data that can be transferred over any communication channel at a given point of time is known as the **bandwidth**. In analog systems, it is measured in hertz (Hz) and in digital systems; it is measured in bits per second (bps).

4. **Data transfer rate:** - The amount of data transferred per second by the communication channel from one point to another is known as the data transfer rate. It is measured in bits per second (bps), bytes per second (Bps).

|                                            |                                             |
|--------------------------------------------|---------------------------------------------|
| <b>1 kbps = 1024 bps ( bit per second)</b> | <b>1 Kbps = 1024 Bps ( Byte per second)</b> |
| 1 mbps = 1024 kbps                         | <b>1 Mbps = 1024 Kbps</b>                   |
| 1 gbps = 1024 mbps                         | <b>1 Gbps = 1024 Mbps</b>                   |
| 1 tbps = 1024 gbps                         | <b>1 Tbps = 1024 Gbps</b>                   |

### Transmission media:

1. **Twisted pair cable:** - It consists of two identical 1 mm thick copper wires insulated and twisted together. The twisted pair cables are twisted in order to reduce crosstalk and electromagnetic induction.

#### Advantages:

- (i) It is easy to install and maintain.
- (ii) It is very inexpensive

#### Disadvantages:

- (i) It is incapable to carry a signal over long distances without the use of repeaters.
- (ii) Due to low bandwidth, these are unsuitable for broadband applications.

2. **Co-axial Cables:** It consists of a solid wire core surrounded by one or more foil or braided wire shields, each separated from the other by some kind of plastic insulator. It is mostly used in the cable wires.

#### Advantages:

- (i) Data transmission rate is better than twisted pair cables.
- (ii) It provides a cheap means of transporting multi-channel television signals around metropolitan areas.

#### Disadvantages:

- (i) Expensive than twisted pair cables.
- (ii) Difficult to manage and reconfigure.

3. **Optical fiber:** - An optical fiber consists of thin glass fibers that can carry information in the form of visible light.

#### Advantages:

- (i) **Transmit data over long distance with high security.**
- (ii) **Data transmission speed is high**
- (iii) Provide better noise immunity
- (iv) Bandwidth is up to 10 Gbps.

#### Disadvantages:

- (i) Expensive as compared to other guided media.
- (ii) Need special care while installation?

4. **Infrared:** - The infrared light transmits data through the air and can propagate throughout a room, but will not penetrate walls. It is a secure medium of signal transmission. The infrared transmission has become common in TV remotes, automotive garage doors, wireless speakers etc.

5. **Radio Wave:** - Radio Wave an electromagnetic wave with a wavelength between 0.5 cm and 30,000 m. The transmission making use of radio frequencies is termed as radio-wave transmission

Advantages:

- (i) Radio wave transmission offers mobility.
- (ii) It is cheaper than laying cables and fibers.
- (iii) It offers ease of communication over difficult terrain.

Disadvantages:

- (i) Radio wave communication is insecure communication.
- (ii) Radio wave propagation is susceptible to weather effects like rains, thunder storms etc.

6. **Microwave Wave:** - The Microwave transmission is a line of sight transmission. Microwave signals travel at a higher frequency than radio waves and are popularly used for transmitting data over long distances.

Advantages:

- (i) It is cheaper than laying cable or fiber.
- (ii) It has the ability to communicate over oceans.

Disadvantages:

- (i) Microwave communication is an insecure communication.
- (ii) Signals from antenna may split up and transmitted in different way to different antenna which leads to reduce to signal strength.
- (iii) Microwave propagation is susceptible to weather effects like rains, thunder storms etc.
- (iv) Bandwidth allocation is extremely limited in case of microwaves.

7. **Satellite link:** - The satellite transmission is also a kind of line of sight transmission that is used to transmit signals throughout the world.

Advantages:

- (i) Area covered is quite large.
- (ii) No line of sight restrictions such as natural mountains, tall building, towers etc.
- (iii) Earth station which receives the signals can be fixed position or relatively mobile.

Disadvantages:-

- (i) Very expensive as compared to other transmission mediums.
- (ii) Installation is extremely complex.
- (iii) Signals sent to the stations can be tampered by external interference.

**Network devices:**

**Modem:** A MODEM (MODulator DEModulator) is an electronic device that enables a computer to transmit data over telephone lines. There are two types of modems, namely, internal modem and external modem.

**RJ45 connector:** - The RJ-45(Registered Jack) connectors are the plug-in devices used in the networking and telecommunications applications. They are used primarily for connecting LANs, particularly Ethernet.

**Ethernet Card:** - It is a hardware device that helps in connection of nodes within a network.

**Hub:** A hub is a hardware device used to connect several computers together. Hubs can be either active or passive. Hubs usually can support 8, 12 or 24 RJ45 ports.

**Switch:** A switch (switching hub) is a network device which is used to interconnect computers or devices on a network. It filters and forwards data packets across a network. The main difference between hub and switch is that hub replicates what it receives on one port onto all the other ports while switch keeps a record of the MAC addresses of the devices attached to it.

**Gateway:** A gateway is a device that connects dissimilar networks.

**Repeater:** A repeater is a network device that amplifies and restores signals for long distance transmission.

### **Network Topologies and types:**

**The BUS Topology:** - The bus topology uses a common single cable to connect all the workstations. Each computer performs its task of sending messages without the help of the central server. However, only one workstation can transmit a message at a particular time in the bus topology.

#### Advantages:

- (i) Easy to connect and install.
- (ii) Involves a low cost of installation time.
- (iii) Can be easily extended.

#### Disadvantages:-

- (i) The entire network shuts down if there is a failure in the central cable.
- (ii) Only a single message can travel at a particular time.
- (iii) Difficult to troubleshoot an error.

**The STAR Topology:** - A STAR topology is based on a central node which acts as a hub. A STAR topology is common in homes networks where all the computers connect to the single central computer using it as a hub.

#### Advantages:

- (i) Easy to troubleshoot
- (ii) A single node failure does not affect the entire network.
- (iii) Fault detection and removal of faulty parts is easier.
- (iv) In case a workstation fails, the network is not affected.

#### Disadvantages:-

- (i) Difficult to expand.
- (ii) Longer cable is required.
- (iii) The cost of the hub and the longer cables makes it expensive over others.
- (iv) In case hub fails, the entire network fails.

**The TREE Topology:** - The tree topology combines the characteristics of the linear bus and the star topologies. It consists of groups of star - configured workstations connected to a bus backbone cable.

#### Advantages:

- (i) Eliminates network congestion.
- (ii) The network can be easily extended.
- (iii) Faulty nodes can easily be isolated from the rest of the network.

#### Disadvantages:

- (i) Uses large cable length.
- (ii) Requires a large amount of hardware components and hence is expensive.
- (iii) Installation and reconfiguration is very difficult.

### **Types of Networks:**

**LAN (Local Area Network):** A Local Area Network (LAN) is a network that is confined to a relatively small area. It is generally limited to a geographic area such as writing lab, school or building. It is generally privately owned networks over a distance not more than 5 Km.

**MAN (Metropolitan Area Network):** MAN is the networks cover a group of nearby corporate offices or a city and might be either private or public.

**WAN (Wide Area Network):** These are the networks spread over large distances, say across countries or even continents through cabling or satellite uplinks are called WAN.

**PAN (Personal Area Network):** A Personal Area Network is computer network organized around an individual person. It generally covers a range of less than 10 meters. Personal Area Networks can be constructed with cables or wirelessly.

### Network Protocol:

- TCP/IP (Transmission Control Protocol / Internet Protocol)
- PPP (Point to Point Protocol)
- FTP (File Transfer Protocol )
- TELNET ( TERminal NETwork OR TELecommunication NETwork)
- GSM ( Global System for Mobile Communications)
- (vi)CDMA( Code Division Multiple Access)
- (vii)GPRS(General Packet Radio Service)
- WLL( Wireless Local Loop)
- SMTP(Simple Mail Transfer Protocol)
- POP3(Post Office Protocol 3)
- VoIP(Voice over Internet Protocol)
- Wi-Fi (Wireless Fidelity)
- WiMAX (Worldwide Interoperability for Microwave Access)
- 1 G: 1 G stands for the first generation of wireless analog cellular.
- 2 G: 2 G (Second Generation) is digital cellular comprising integrated voice and data communication.

3 G: Some of the advantages of 3 G are:-

- (i) Broadband capabilities, greater power and capacity.
- (ii) Higher data rate at lower cost than 2G.
- (iii) High speed data transmission
- (iv) Global roaming, wider coverage.

### Network Security Concepts:

**Viruses:** Viruses are programs which replicate and attach to other programs in order to corrupt the executable codes. Virus enters the computer system through an external source and become destructive.

**Worms:** Worms are also self- replicating programs that do not create multiple copies of itself on one computer but propagate through the computer network. Worms log on to computer systems using the username and passwords and exploit the system.

**Trojan horse:** - Though it is a useful program, however, a cracker can use it to intrude the computer system in order to exploit the resources. Such a program can also enter into the computer through an e-mail or free programs downloaded through the Internet.

**Spams:** Unwanted e-mail (usually of a commercial nature sent out in bulk)

**Cookies:** Cookies are the text messages sent by a web server to the web browser primarily for identifying the user.

**Firewall:** A firewall is used to control the traffic between computer networks. It intercepts the packets between the computer networks and allows only authorized packets to pass.

**Cyber Law:** Cyber law refers to all the legal and regulatory aspects of Internet and the World Wide Web.

**Cyber Crimes:** Cyber crime involves the usage of the computer system and the computer network for criminal activity.

**Hacking:** Hacking is an unauthorized access to computer in order to exploit the resources.

### Web Services:

**WWW:** The World Wide Web or W3 or simply the Web is a collection of linked documents or pages, stored on millions of computers and distributed across the Internet.

**HTML (Hyper Text Markup Language):-** HTML is a computer language that describes the structure and behavior of a web page. This language is used to create web pages.

**XML (eXtensible Markup Language):-** Extensible Markup Language (XML) is a meta language that helps to describe the markup language.

**HTTP (Hyper Text Transfer Protocol):-** A protocol to transfer hypertext requests and information between servers and browsers.

**Domain Names:** A domain name is a unique name that identifies a particular website and represents the name of the server where the web pages reside.

**URL:-** The Uniform Resource Locator is a means to locate resources such as web pages on the Internet. URL is also a method to address the web pages on the Internet. There are two types of URL, namely, absolute URL and relative URL.

**Website:** A collection of related web pages stored on a web server is known as a website.

**Web browser:** A software application that enables to browse, search and collect information from the Web is known as Web browser.

**Web Servers:** The web pages on the Internet are stored on the computers that are connected to the Internet. These computers are known as web servers.

**Web Hosting:** - Web Hosting or website hosting is the service to host, store and maintain the websites on the World Wide Web.

**Web Scripting:** - The process of creating and embedding scripts in a web page is known as Web Scripting. Types of Scripts:-

- (i) **Client Side Scripts:** - Client side scripts supports interaction within a webpage. E.g. VBScript, Java Script, PHP (PHP" S Hypertext Preprocessor).
- (ii) **Server Side Scripts:** - Server side scripting supports execution at server - end. E.g. ASP, JSP, PHP

### Open Source Concepts:

**Open Source Software:** - Software whose source code is available and which can be modified copied and redistributed. It may be free of cost or not.

**Freeware:** - The software that is free of cost and can be copied redistributed but can" t be modified because source code is not available.

**Shareware:** - Software for which license fee is payable after some time limit.

**Proprietary Software:** - Software that is neither free nor open.

Difference among OSS, Free Software and Freeware:-

| S. No. | OSS(Open Source Software)       | Free Software                   | Freeware                                      |
|--------|---------------------------------|---------------------------------|-----------------------------------------------|
| 1.     | May be free of cost or not      | Free of cost                    | Free of cost                                  |
| 2.     | Source code available           | Source code available           | Source code not available                     |
| 3.     | Modified, copied, redistributed | Modified, copied, redistributed | Copied, redistributed but can" t be modified. |

**FLOSS:** - FLOSS refers to Free Libre and Open Source Software or to Free Livre and Open Source Software. The term FLOSS is used to refer to software which is both free software as well as open source software.

**FOSS:** - software which is free as well as open belongs to category FOSS (Free and Open Source Software).

**GNU:** - GNU is recursive acronym for GNU's Not Unix. The GNU project was launched in 1984 to develop a complete UNIX like operating system which is free software. GNU project expanded and now it is not limited to an operating system but also includes application part.

**FSF:** - Free Software Foundation is a non-profit organization created for the purpose of supporting free software environment. It was founded by Richard Stallman in 1985 to support GNU Project and GNU licenses.

**OSI:** - Open Source Initiative is an organization dedicated to cause of promoting open source software. It was founded by Bruce Perens and Eric Raymond in Feb. 1998.

**Network Design:** - The aim of the network design is to minimize the load on the network backbone. The 80-20 rule helps to build a good network design. This rule states that in a LAN, 80 percent traffic should be local and 20 percent traffic should be allowed across the backbone.

### Tips to solve Questions based on Networking

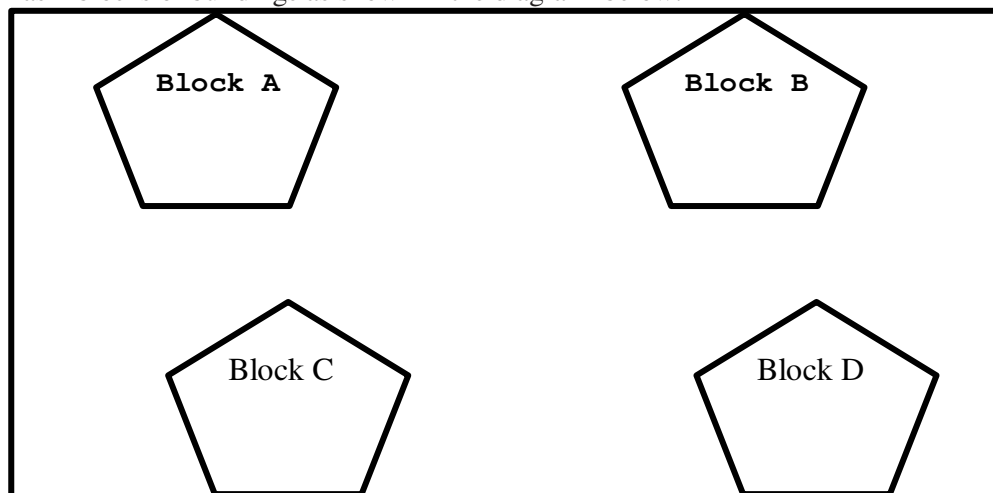
1. **Where Server should be placed:** Server should be placed in the building where the number of computers is **maximum**.
2. **Suggest a suitable cable layout of connection:** A suitable cable layout can be suggested in the following two ways:-
  - (i) **On the Basis of Server:** First the location of the Server is found out. Server is placed in that building where the number of computers are maximum (According to 80 – 20 rule). After finding the server position, each building distance is compared with the Server building directly or indirectly (taking other building in between). The shortest distance is counted whether it is through directly or indirectly.
  - (ii) **On the Basis of Distance from each building:** The distance between the each building is compared to all other buildings either directly or indirectly. The shortest distance is counted whether it is directly or through some other building.
3. Where the following devices be placed:
  - (i) **MODEM:-**
  - (ii) **HUB / SWITCH:-**
  - (iii) **BRIDGE:**
  - (iv) **REPEATER:** It is used if the distances higher than 70 m. It regenerates data and voice signals.
  - (v) **ROUTER:** When one LAN will be connected to the other LAN.

### UNSOLVED PROBLEMS

- Problem 1: Differentiate between Internet and Intranet.
- Problem 2: Define the following switching techniques:
  - (i) Circuit Switching
  - (ii) Message Switching
  - (iii) Packet Switching
- Problem 3: Define the term Bandwidth. Give unit of Bandwidth.
- Problem 4: Write two advantages and two disadvantages of the following Transmission media:
  - (i) Twisted Pair
  - (ii) Co-axial
  - (iii) Optical Fiber
  - (iv) Radio Waves
  - (v) Microwave Waves



- (vi) Satellite link
- Problem 5: Define the following Network devices:
- (i) MODEM
  - (ii) Hub
  - (iii) Switch
  - (iv) Gateway
- Problem 6: Write two advantages and two disadvantages of the following Network Topologies:-
- (i) STAR
  - (ii) BUS
  - (iii) TREE
- Problem 7: Define the following types of Network:
- (i) LAN
  - (ii) MAN
  - (iii) WAN
- Problem 8: Define the following Network Security Concepts:
- (i) Viruses
  - (ii) Worms
  - (iii) Trojan horse
  - (iv) Spams
- Problem 9: What do you understand by the terms Cookies and Firewall?
- Problem 10: What is significance of Cyber Law?
- Problem 11: How is a Hacker different from a Cracker?
- Problem 12: Expand the following terms:  
FLOSS, FOSS, GNU, FSF, OSI, HTML, XML, HTTP, URL, PHP, ASP, JSP, TCP / IP, FTP, PPP, GSM, CDMA, WLL, 3G, SMS, LAN, MAN, WAN, W3C, SMTP, POP, Wi-Fi
- Problem 13: Compare and Contrast the following:-
- (i) Free Software and Open Source Software
  - (ii) OSS and FLOSS
  - (iii) Proprietary software and Free software.
  - (iv) Freeware and Shareware
  - (v) Freeware and Free software.
- Problem 14: ABC Corporation has set up its new center at Delhi for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below:



**Center to center distances between various blocks**

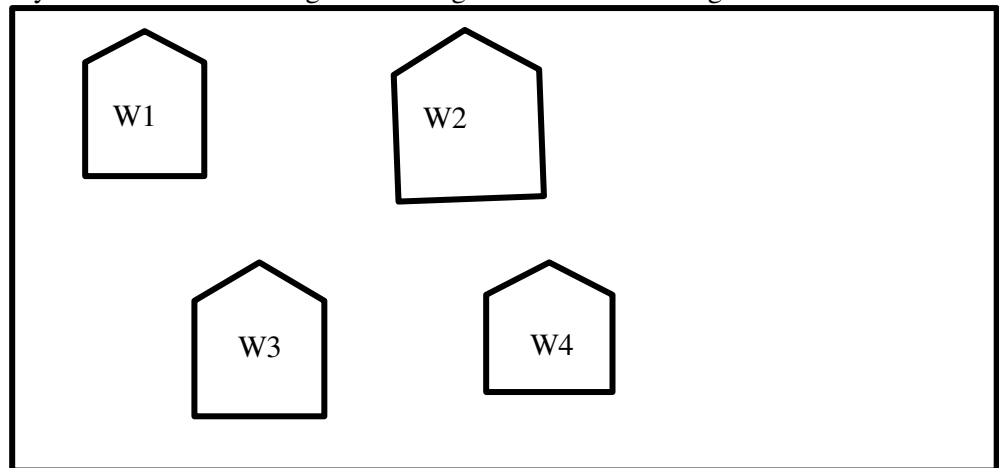
- Block A to Block B 50 m
- Block B to Block C 150 m
- Block C to Block D 25 m
- Block A to Block D 170 m
- Block B to Block D 125 m
- Block A to Block C 90 m

**Number of Computers**

Block A 25  
 Block B 50  
 Block C 125  
 Block D 10

- Suggest a cable layout of connections between the blocks.
- Suggest the most suitable place (i.e. block) to house the server of this organization with a suitable reason.
- Suggest the placement of the following devices with justification
  - Repeater
  - Hub/Switch
- The organization is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

Problem 15: A company in Reliance has 4 wings of buildings as shown in the diagram:



Center to center distances between various Buildings:

|          |      |
|----------|------|
| W3 to W1 | 50m  |
| W1 to W2 | 60m  |
| W2 to W4 | 25m  |
| W4 to W3 | 170m |
| W3 to W2 | 125m |
| W1 to w4 | 90m  |

Number of computers in each of the wing:

|    |     |
|----|-----|
| W1 | 150 |
| W2 | 15  |
| W3 | 15  |
| W4 | 25  |

Computers in each wing are networked but wings are not networked. The company has now decided to connect the wings also.

- Suggest a most suitable cable layout & topology of the connection between the wings.
- The company wants internet accessibility in all the wings. Suggest an economic technology .
- Suggest the placement of the following devices with justification if the company wants minimized network traffic :
  - Repeater
  - Hub
  - Switch

## Higher Order Thinking Skills (HOTS) Questions

Q 1: WHAT WILL BE OUTPUT OF FOLLOWING PROGRAM?

```
#include<iostream.h>
include <conio.h>
void main()
{
clrscr();
int sum(int(*) (int),int);
int square(int);
int cube(int);
cout<<sum(square,4)<<endl;
cout<<sum(cube,4)<<endl;
getch();
}
int sum(int(*ptr) (int k),int n)
{
int s=0;
for(int i=1;i<=n;i++)
{
s+=(*ptr) (i);
}
return s;
}
int square(int k)
{ int sq;
sq=k*k;
return k*k;
}
int cube(int k)
{
return k*k*k;
}
```

**ANS 1: OUTPUT WILL BE**

30  
100

Q2 How many times will the following program will print "examination"?

```
#include<iostream.h>
void main()
{
while(1)
{
cout<<"examination"
}
}
```

**ANS 2:Unless ^C is pressed ,program will print "examination" infinitely.**

Q3:What would be contents of following after array initialization?

```
int A[5]={3,8 ,9}
```

**Ans 3:**

**A**

|   |   |   |   |   |
|---|---|---|---|---|
| 3 | 8 | 9 | 0 | 0 |
|---|---|---|---|---|

Q 4: What is the difference between the constructor and normal function?

Ans4.

| Constructor                                                      | Normal Function                                                  |
|------------------------------------------------------------------|------------------------------------------------------------------|
| 1. Constructor has same name as class name.                      | 1. A normal function can have any legal name but not class name. |
| 2. Constructor can not have any return type value not even void. | 2. A function should have any return type value.                 |
| 3. Constructor is automatically called.                          | 3. A function is explicitly called.                              |
| 4. Constructor can not be static.                                | 4. A Function can be static.                                     |

Q 5: What is the similarity between class and the constructor? (HOTS)/Bright Student

Ans 5: The only similarity between constructor and is that constructor has same name as class name.

Q 6: Find the output of the following program?

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class state
{ char *statename;
 int size;
public:
 state(){size=0;statename=new char[size+1];}
 state (char *s)
 { size=strlen(s);statename=new char[size+1];
 strcpy(statename,s);
 }
 void display()
 { cout<<statename<<endl;}
 void replace(state&a, state &b)
 {size=a.size+b.size;
 delete statename;
 statename=new char[size+1];
 strcpy(statename, a.statename);
 strcat(statename,b.statename);
 }
};
void main()
{ clrscr();
 char *temp="Delhi";
 state state1(temp), state2("Mumbai"), state3("Nagpur"), s1,s2;
 s1.replace(state1,state2);
 s2.replace(s1,state3);
 s1.display();
 s2.display();
 getch();
}
```

**Ans 6: DelhiMumbai**

**DelhiMumbaiNagpur**

Q7: Find out errors in the following program:-

```
class number
{
 int x=10;
 float y;
 number(){ x=y=10;}
public:
 number(number t)
 {
 x=t.x; y=t.y;
 }
 ~ (){ cout<<"Object destroyed ";}
}
main()
```

```
{
 number a1, a2(a1);
}
```

**Ans 7: error: int x=10; // class member can not be initialized in the class.  
 Constructor should be declared in public section of class.  
 Reference operator is missing in the definition of copy constructor  
 In destructor class name is missing.  
 Semicolon is missed after the definition of class.**

Q 8: What is the difference between nesting or containership and inheritance?  
 Explain with example?

Ans 8: Containership or Nesting: When a class contains object of other class type as its data member is known as containership or nesting.

Inheritance: Inheritance is the process of creating new class by reusing the properties of an existing class by accessing them depending on different visibility mode. The new class is called derived and existing class is called base class.

Q 9: What will be the output of the program?

```
#include<iostream.h>
class base
{ public:
 void display()
 {
 cout<<"It is a base class "<<endl;
 }
};
class derived: public base
{
 public:
 void display()
 { cout<<"It is a derived class "<<endl;}
};
main()
{
 derived ob1;
 ob1.display();
}
```

**Ans 9: The output will be:  
 It is a derived class.**

Q 10: Define a class named Tour in C++ with following description?  
 Private members:

tcode                    integer (Ranges 6 - 10)  
 adults, children, distance    integer  
 totalfare                float

AssignFare( )    A function which calculates and assign the value to data member totalfare as follows:-

- **For adults**                Fare                                Distance  
                                      Rs. 500                                >=1500

And fare get reduced by 25% if distance is < 1500.

- **For Children**

For every child a fixed Rs. 50 is charged as fare.

Public members:

- A constructor which initialized initialize all data members with 0
- Function EnterTour() to input the values of the data members tcode, adults, children and call to AssignFare function.
- Function ShowTour() to print all the details of object of Travel type.

**Ans 10:**

```
class tour
{
 int tcode,adults,children,distance;
 float totalfare;
 void assignfare()
```

```

{
 float cfare=50, afaire=1500;
 if(distance<1500)
 afaire=afaire-(afaire*25/100);
 totalfaire=(children*cfare)+(adults*afaire);
}
public:
travel()
{
 tcode=adults=children=distance=totalfaire=0; }
void entertour()
{
 do
 {
 cout<<"Enter tcode between 6-10 ";
 cin>>tcode;
 if (tcode<6 || tcode>10)
 cout<<"Invalid tcode "<<endl;
 }while(tcode<6 || tcode>10);
 cout<<"Enter children, adults, distance";
 cin>>children>>adults>>distance;
 assignfaire();
}
void showtour()
{
 cout<<"tcode:"<<tcode<<endl;
 cout<<"children:"<<children<<endl;
 cout<<"adults :"<<adults<<endl;
 cout<<"distance:"<<distance<<endl;
 cout<<"total faire:"<<totalfaire<<endl;
}
};

```

Q 11: Define a class named Admission in C++ with following description?

**Private members:**

admno            integer (Ranges 10-1500)  
name             string of 20 characters  
cls               integer  
fees              float

**Public members:**

A constructor which initialized admno with 10, name with "NULL", cls with 0 & fees with 0

Function getdata() to read the object of Admission type.

Function putdata() to print the details of object of admission type.

Function draw\_nos() to generate the admission no. randomly to match with admno and display the detail of object.

```

Ans 11:
class admission
{
 int admno;
 char name[20];
 int cls;
 float fees;
public:
 admission()
{
 admno=10;
 strcpy(name,"NULL");
 cls=0;
 fees=0;
}
void getdata()
{
 do
 {
 cout<<"Enter admno between 10-1500 ";
 cin>>admn
 if (admno<10 || admno>1500)
 cout<<"Invalid admission no !"<<endl;
 }while(admno<10 ||admno>1500);
 cout<<"Enter name ";
 gets(name);
 cout<<"Enter class and fees ";
 cin>>cls>>fees;
}
}

```

```

 }
 void putdata()
 {
 cout<<"Admno : "<<admno<<endl;
 cout<<"Name : "<<name<<endl;
 cout<<"Class : "<<cls<<endl;
 cout<<"Fees : "<<fees<<endl;
 }
 void draw_nos()
 {
 int num;
 randomize();
 num=random(1491)+10;
 if (num==admno)
 putdata();
 }
};
Q 12:
Class testmeout
{
 int rollno;
public:
 ~testmeout() //Function 1
 {
 cout<<rollno<<" is Leaving examination hall"<<endl;
 }
 testmeout() //Function 2
 {
 rollno=1;
 cout<<rollno<<" is appearing for examination "<<endl;
 }
 testmeout(int n, char name[]) //Function 3
 {
 rollno=n;
 cout<<name<<" is in examination hall"<<endl;
 }
 testmeout(testmeout & t); //function 4
 void mywork() //Function 5
 {
 cout<<rollno<<" is attempting questions "<<endl;
 }
};

```

- i) In object oriented programming, what is Function 1 referred as and when does it get invoked?
  - ii) In object oriented programming, what is Function 2 referred as and when does it get invoked?
  - iii) In object oriented programming, what is Function 3 referred as and when does it get invoked?
  - iv) Write a statement so that function 3 gets executed?
- Complete the definition of function 4
- v) What will be the output of the above code if its main function definition is as given below (assumed the definition of Function 4 is completed) :

```

main()
{testmeout obl;
 obl.mywork();
}

```

- vi) Which feature of object oriented programming is demonstrated using Function 2, Function 3 and Function 4 in the above class testmeout?
- vii) What is the scope of data member (rollno) of class testmeout? What does the scope of data members depend upon?

**Ans 12:**

- i) It is referred as destructor. It is automatically invoked when an object of concerned class goes out of scope.
- ii) It is referred as constructor. It is automatically invoked when an object of concerned class is declared / created.
- iii) It is parameterized constructor and gets invoked when an object of concerned class is created / declared with the matched parameters.
- iv) testmeout obl(15, "Vicky");  
testmeout (testmeout & t) { rollno=t.rollno;}
- v) output will be :  
1 is appearing for examination

1 is attempting questions

1 is Leaving examination hall

vi) It is constructor overloading. It shows Polymorphism feature of the OOP.

vii) The rollno member of object can only be used by the concerned object where that object is declared. Its scope basically depends upon the concerned object.

Q 13: Given two arrays of integers A and B of sizes M and N respectively. Write a function named MIX() which will produce a third array named C, such that the following sequence is followed :

All even numbers of A from left to right are copied into C from left to right.

All odd numbers of A from left to right are copied into C from right to left

All even numbers of B from left to right are copied into C from left to right.

All odd numbers of B from left to right are copied into C from right to left

A, B and C are passed as arguments to MIX().

e.g. : A is {3,2,1,7,6,3} and B is {9,3,5,6,2,8,10}, the resultant array C is {2,6,6,2,8,10,5,3,9,3,7,1,3}

```

Ans 13: void mix (int A[], int B[], int n, int m)
 {
 int c[20],i=0,j=0,k=0,l;
 l=m+n-1;
 while (i<n && k<20)
 {
 if (A[i]%2==0)
 C[k++] = A[i++];
 else C[l--] = A[i++];
 }
 While (j<m && k<20)
 {
 if (B[j]%2==0)
 C[k++]=B[j++];
 else C[l--]=B[j++];
 }
 cout<<" \n\nThe elements of an array C is :";
 for (i=0;i<m+n;i++)
 cout<<"\n"<<C[i];
 }
void main()
 {
 int A[j]= { 3,2,1,7,6,3}, B[]= {9,3,5,6,2,8,10};
 Mix(A,B,6,7);
 }

```

Q 14. Suppose an array P containing float is arranged in ascending order. Write a user defined function in C++ to search for one float from P with the help of binary search method. The function should return an integer 0 to show absence of the number and integer 1 to show presence of the number in the array. The function should have the parameters as (1) an array (2) the number DATA to be searched (3) number of element N.

```

Ans 14: int bsearch (float P[10], float DATA, int N)
 {
 int beg =0, end = N-1,mid, pos = -1;
 while(beg<=end)
 {
 mid = (beg+ end)/2;
 if (P[mid] == DATA)
 {
 pos =mid +1;
 Break;
 }
 else if (item > AE[mid])
 beg = mid +1;
 else
 end = mid-1;
 }
 return ((pos== -1)? 0:1);
 }

```

Q 15: Write a function in C++ to perform a PUSH operations on a dynamically allocated stack containing real number?

```

Ans 15:
 struct Node
 {

```



```

 float data;
 Node * next;
 };
Void push (Node*Top, float num)
{
 Node*nptr = new Node;
 nptr -> data = num;
 nptr -> next = NULL;
 if(Top == NULL)
 Top = nptr;
 else
 {
 nptr -> next = Top;
 Top = nptr;
 }
}

```

Q 16: Each node of a STACK containing the following information, in addition to required pointer field:

Roll no. of the student

Age of the student.

Gve the structure of node for the linked stack in question.

TOP is a pointer to the topmost node of the STACK. Write the following function:

PUSH() - TO push a node in to the stack which is allocated dynamically.

POP() - Te remove a node from the stack and to release the memory.

**Ans 16:**

```

struct STACK
{
 int rollno, age;
 STACK*next;
} *top, *nptr, *ptr;
void pop()
{
 if (!top) { cout << "\nUnderflow!!" ; exit(1); }
 else
 {
 cout << '\n' << top -> rollno << '\t' << top -> age;
 ptr = top;
 top = top -> next;
 delete ptr;
 }
}
void push()
{
 nptr = new stack; //allocate memory
 cout << "\n Enter roll number and age to be inserted : " ;
 cin >> nptr-> rollno >> nptr->age ;
 nptr -> next = NULL;
 if (!top) top = nptr;
 else
 {
 ptr -> next = top;
 top = nptr
 }
}

```

Q 17: Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of the middle row and the elements of middle column.

Example if the array content is

```

3 5 4
7 6 9
2 1 8

```

Output through the function should be:

Middle row: 769                      Middle column: 5 6 1

**Ans 17:**

```

// Function to display the elements which lie on middle of row and column
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
const M = 10;
const N = 10;
void display_RowCol(int Array[M][N], int r, int c)
{
 int row = r / 2;
 int col = c / 2;
 // Finding the middle row
 cout << "Middle Row : ";
 for(int j=0; j<c; j++)
 cout << Array[row][j] << " ";
 cout << endl;
 // Finding the middle column
 cout << "Middle Column : ";
 for(j=0; j<c; j++)
 cout << Array[j][col] << " ";
 getch();
}
void main()
{
 int Array[M][N];
 int i, j;
 int r, c;
 cout << "Enter total no. of rows: ";
 cin >> r;
 cout << "Enter total no. of columns: ";
 cin >> c;
 if ((r == c) && ((r%2==1) && (c%2==1)))
 {
 cout << "Input steps";
 cout << "\n\nEnter the element in the array\n";
 for(i=0; i<r; i++)
 for(j=0; j<c; j++)
 {
 cin >> Array[i][j];
 }
 }
 else
 {
 cout << "Input row and column not valid";
 getch();
 return;
 }
 display_RowCol(Array, r, c);
}

```

Q 18: Define function stackpush( ) to insert nodes and stack pops ( ) to delete nodes for a linked list implemented stack having the following structure for each node

```

struct Node
{
 Char name [20]
 Int age ;
 Node * link ;
};
Class stuck {
 Node * top ;
 Public
 Stack () { top = null ;} ;
 Void stackpush () ;
 Void stackpop () ;
}

```

Ans 18:

```

void stack::stackpush()
{
 int val;
 node *temp;
 temp = new node;
}

```

```

cout << "Enter name : "; gets(temp-
>name);
cout << "Enter age : ";
cin >> temp->age;
temp->link = NULL;
if(top ==NULL)
 top = temp;
else
{
 temp->link = top;
 top = temp;
}
}

void stack::stackpop()
{
 node *temp;
 if (top == NULL)
 {
 cout << "Stack Empty ";
 }
 else
 {
 temp = top;
 top = top->link;
 temp->link = NULL;
 delete temp;
 }
}

```

Q 19: Assuming the class Vehicle as follows:

```

Class vehicle
{ char vehicletype[10];
int no_of_wheels;
public:
 void getdetials()
 { gets(vehicletype);
 cin>>no_of_wheels;
 }
 void showdetails()]
 { cout<<"Vehicle Type"<<vehicletype;
 cout<<"Number of Wheels="<<no_of_wheels;
 }
}

```

Write a function showfile() to read all the records present in an already existing binary file SPEED.DAT and display them on the screen ,also count the number of records present in the file.

**Ans 19:**

```

void showfile()
{
 ifstream fin;
 fin.open("SPEED.DAT",ios::in|ios::binary);
 vehicle v1;
 int count=0;
 while (!fin.eof())
 {
 fin.read((char *)&v1,sizeof(v1));
 count++;
 v1.showdetails();
 }
 cout<<"Total number of records are "<<count;
}

```

Q 20: What are DDL and DML?

**Ans:-** The DDL provides statements for the creation and deletion of tables and indexes.

The DML provides statements to enter, update, delete data and perform complex queries on these tables.

Q 21: Write the rules to name an objects?

**Ans :**

- The maximum length must be 30 character long.
- The Object name should not contain quotation mark.
- The name must start with letter.
- The use of \$ and # is discouraged in the object name.
- A name must not be a reserved name.

Q 22: Write the SQL query commands based on following table

**TABLE: GRADUATE**

| S.NO | NAME    | STIPEND | SUBJECT   | AVERAGE | DIV. |
|------|---------|---------|-----------|---------|------|
| 1    | KARAN   | 400     | PHYSICS   | 68      | I    |
| 2    | DIWAKAR | 450     | COMP. Sc. | 68      | I    |
| 3    | DIVYA   | 300     | CHEMISTRY | 62      | I    |
| 4    | REKHA   | 350     | PHYSICS   | 63      | I    |
| 5    | ARJUN   | 500     | MATHS     | 70      | I    |
| 6    | SABINA  | 400     | CEHMISTRY | 55      | II   |
| 7    | JOHN    | 250     | PHYSICS   | 64      | I    |
| 8    | ROBERT  | 450     | MATHS     | 68      | I    |
| 9    | RUBINA  | 500     | COMP. Sc. | 62      | I    |
| 10   | VIKAS   | 400     | MATHS     | 57      | II   |

- (a) List the names of those students who have obtained **DIV I** sorted by NAME.
- (b) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.
- (c) To count the number of students who are either PHYSICS or COMPUTER SC graduates.
- (d) To insert a new row in the GRADUATE table:  
11,"KAJOL", 300, "COMP. SC.", 75, 1
- (e) Give the output of following sql statement based on table GRADUATE:
- (i) Select MIN(AVERAGE) from GRADUATE where SUBJECT="PHYSICS";
- (ii) Select SUM(STIPEND) from GRADUATE WHERE div=2;
- (iii) Select AVG(STIPEND) from GRADUATE where AVERAGE>=65;
- (iv) Select COUNT(distinct SUBDJECT) from GRADUATE;
- Assume that there is one more table GUIDE in the database as shown below:

**Table: GUIDE**

| MAINAREA    | ADVISOR |
|-------------|---------|
| PHYSICS     | VINOD   |
| COMPUTER SC | ALOK    |
| CHEMISTRY   | RAJAN   |
| MATHEMATICS | MAHESH  |

- (f) What will be the output of the following query:  
SELECT NAME, ADVISOR FROM GRADUATE, GUIDE WHERE SUBJECT= MAINAREA;

**Ans :**

- (a) SELECT NAME FROM GRADUATE WHERE DIV='I' ORDER BY NAME;
- (b) SELECT NAME, STIPEND, SUBJECT, STIPEND\*12 STIPEND\_YEAR FROM GRADUATE;
- (c) SELECT SUBJECT, COUNT(NAME) FROM GRADUATE GROUPBY (SUBJECT) HAVING SUBJECT='PHYSICS' OR SUBJECT='COMP. Sc.';
- (d) INSERT INTO GRADUATE VALUES (11, 'KAJOL', 300, 'COMP. Sc.', 75, 1);
- (e) (i) MIN(AVERAGE) 63  
(ii) SUM(STIPEND) 800  
(iii) AVG(STIPEND) 420  
(iv) COUNT(DISTINCTSUBJECT) 4
- (f) SELECT NAME, ADVISOR FROM GRADUATE, GUIDE WHERE SUBJECT=MAINAREA;

|        |         |
|--------|---------|
| NAME   | ADVISOR |
| DIVYA  | RAJAN   |
| SABINA | RAJAN   |
| KARAN  | VINOD   |
| REKHA  | VINOD   |
| JOHN   | VINOD   |

Q 23: Prove that  $X \cdot (X+Y) = X$  by algebraic method.

Ans 23:

$$\begin{aligned}
 \text{L.H.S.} &= X \cdot (X+Y) = X \cdot X + X \cdot Y \\
 &= X + X \cdot Y \\
 &= X \cdot (1+Y) \\
 &= X \cdot 1 = X = \text{R.H.S}
 \end{aligned}$$

Q 24: Give duals for the following :

- a)  $A + \bar{A}B$   
 b)  $AB + \bar{A}B$

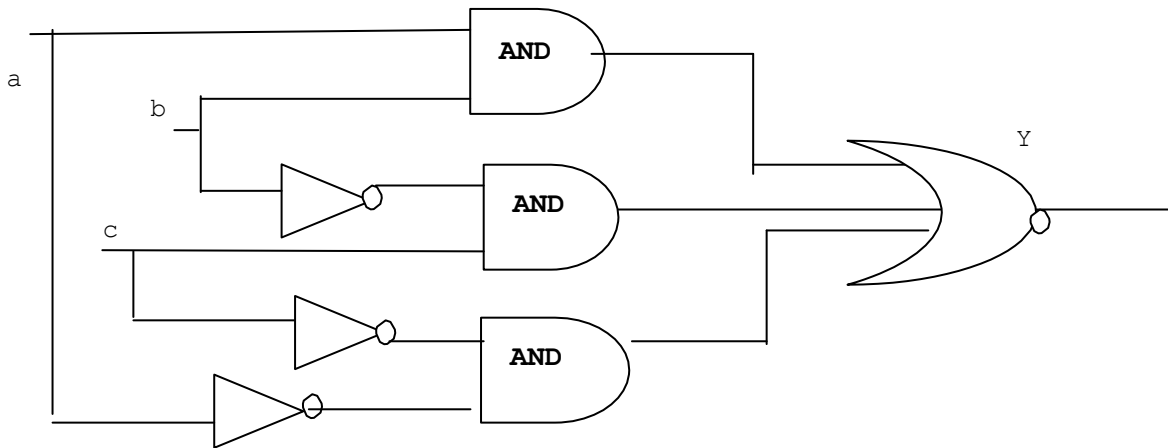
Ans 24:

- a)  $A \cdot (\bar{A} + B)$                       b)  $(A + B) \cdot (\bar{A} + B)$

Q 25: Draw logic circuit diagram for the following expression:

$$Y = \bar{A}B + BC + C\bar{A}$$

Ans 25:



Q 26: What is the difference between baseband and broadband transmission?

Ans 26: Baseband is a bi-directional transmission while broadband is a unidirectional transmission.

No Frequency division multiplexing possible in base band but possible in broadband.

| SNo | Baseband                                              | Broadband                                                                                                              |
|-----|-------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| 1   | Entire bandwidth of the cable is consumed by a signal | broadband transmission, signals are sent on multiple frequencies, allowing multiple signals to be sent simultaneously. |
| 2   | Digital signals                                       | Analog signals                                                                                                         |
| 3   | bi-directional transmission                           | unidirectional transmission                                                                                            |
| 4   | No Frequency division multiplexing possible           | Frequency division multiplexing possible                                                                               |
| 5   | Uses for short distance                               | Uses for long distance                                                                                                 |

Q 27: What are the difference between domain and workgroup?

Ans 27:

| SNo | Domain                                                                                 | Workgroup                                       |
|-----|----------------------------------------------------------------------------------------|-------------------------------------------------|
| 1.  | One or more computers are servers                                                      | All Computers are peers.                        |
| 2.  | If you have a user account on the domain, you can logon to any computer on the domain. | Each computer has a set of accounts.            |
| 3.  | There can be 100+ computers                                                            | Typically not more then 20-30 computers         |
| 4.  | The computers can be on different local network                                        | All computers must be on the same local netork. |

**SAMPLE PAPER 2010-2011**  
**SUBJECT: COMPUTER SCIENCE (083)**  
**Class: XII**

*Time Allowed: 3 hours*

*Maximum Marks: 70*

Instructions:

- (i) All questions are compulsory.
- (ii) Programming Language: C++

1. (a) What is the purpose of using a **typedef** command in C++? Explain with suitable example. 2
- (b) Write the names of the header files to which the following belong: 1
- (i) abs()
  - (ii) gotoxy()
- (c) Rewrite the following program after removing the syntactical errors (if any). Underline each correction. 2
- ```
include<iostream.h>
include<stdio.h>
class Candidate
{
  int CandidateId=501;
  char Name[20];
public
  Candidate() { }
  void Register() {cin>>CandidateId; gets(Name);}
  void List() {cout<<CandidateId<<": "<<Name<<endl;}
};
void main()
{
  Candidate C;
  Register.C();
  C.List();
}
```
- (d) Find the output of the following program: 3
- ```
#include<iostream.h>
void main()
{clrscr();
 int A[] = {78,80,83,83};
 int *p = A;
 for(int J = 1; J <= 3; J++)
 {cout << *p << "#";
 p++; }
 cout<<endl;
 for(J = 1; J <= 4; J++)
 {(*p)* = 3;
 --p; }
 for(J = 1; J < 5; J++)
 cout<<A[J-1]<<"@";
 cout<< endl;
}
```
- (e) Find the output of the following program: 2
- ```
#include<iostream.h>
#include<string.h>
```

```

#include<ctype.h>
void Secret(char Msg[],int N);
void main()
{char SMS[] = "PreBoard";
  Secret(SMS,1);
  cout<<SMS<<endl;
}
void Secret(char Msg[], int N)
{
  for(int I=0; Msg[I]!='\0'; I++)
    if(I%2==0)
      Msg[I] = Msg[I] + N;
    else
      if(islower(Msg[I]))
        Msg[I] = toupper(Msg[I]);
      else
        Msg[I] = Msg[I] - N;
}

```

- (f) In the following program, if the value of N given by the user is 40, what maximum and minimum values the program could possibly display? 2

```

#include<iostream.h>
#include<stdlib.h>
void main()
{
  randomize();
  int Num,GuessNum;
  cin>>Num;
  GuessNum = random(Num-10) + 41;
  cout<<GuessNum<<endl;
}

```

2. (a) Differentiate between Constructor and Destructor function in context of Classes and Objects using suitable C++ example. 2
- (b) Answer the questions (i) and (ii) after going through the following program: 2

```

#include<iostream.h>
#include<string.h>
class Product
{ char Category[20];
  char Type[20];
  int Quality;
  float Price;
  Product() //Function 1
  {strcpy(Category, "Computer");
   strcpy(Type, "Notebook");
   Quantity = 20;
   Price = 35000;
  }
public:
  void display() //Function 2
  {cout<<Category<<"-"<<Type<<":"<<Quantity
   <<"@"<<Price<<endl;
  }
};
void main()
{
  Product P; //Statement 1
  P.display(); //Statement 2
}

```

- (i) Will statement 1 initialize all the data members for object P with the values given in the Function 1? (Yes or No). Justify your answer. Suggest the correction(s) to be made in the above code.
- (ii) What shall be the possible output when the program gets executed, if the suggested corrections are made in the program?
- (c) Define a class SCHOOLFEE with the following description: 4

Private Members:

SRNumber of type int
 StudentName of type string
 Category of type string
 Subject of type string
 Fee of type float

A function CALC_FEE which calculates and assigns the value of Fee as follows:

For the value of Category as "Boys"

<u>Subject</u>	<u>Fee (Rs.)</u>
ComputerScience	2000
Biology	1500

For the value of Category as "Girls" the above mentioned fee gets reduced by 20%.

Public Members:

A function REGISTER() to input the values of data members SRNumber, StudentName, Category, Subject and invoke the CAL_FEE() function.

A function DISPLAY() which displays the content of all the data members for SCHOOLFEE.

- (d) Answer the questions (i) to (iv) based on the following code: 4

```
class Cricketer
{ char category[20];
protected:
float match_fee;
void calc_match_fee(float);
public:
Cricketer();
void CInput();
void CShow();
};
class Bowler : public Cricketer
{ char BOWLERName[20];
int Wickets;
public:
Bowler();
void BOWInput();
void BOWShow();
};
class Batsman : public Cricketer
{ char BASTSMANName[20];
int Centuries;
float Bat_Average;
public:
Batsman();
void BATInput();
void BATShow();
};
```

- (i) Which type of Inheritance is shown in the above example?
- (ii) How many bytes will be required by an object of the class Batsman?

- (iii) Write name of all the data members accessible from member functions of the class Bowler?
- (iv) Write the name of all the member functions accessible by an object of the class Batsman?
3. (a) Write a function in C++, which accepts an integer array and its size as parameters and sort the array in descending order using insertion sort. 3
- (b) An array Array[40][10] is stored in the memory along with the column with each element occupying 4 bytes. Find out the address of the location Array[30][10] if the location Array[3][6] is stored at the address 8252. 3
- (c) Write a function in C++ to perform a PUSH operation in a dynamically allocated stack considering the following: 3
- ```

struct Node
{ int X, Y;;
 Node *Link;
};
class STACK
{ Node *Top;
 public:
 STACK() {TOP = NULL;}
 void PUSH();
 void POP();
 ~STACK();
};

```
- (d) Write a function in C++ to find the sum of each diagonal in a 2D array of size 3X3. 3
- (e) Convert the following infix expression to its equivalent postfix expression showing stack contents for the conversion: 2  
 $A + B * (C - D) / E$
4. (a) Distinguish between ios::out and ios::app modes. 1
- (b) Write a function TOTAL\_VOWELS( ) in C++ to count the total number of vowels present in a text file "VOWELS.TXT". 2
- (c) Given a binary file SPORTS.DAT, containing records of the following structure type: 3
- ```

struct sports
{ char Event[20];
  Char Participant[10][30];
};

```
- Write a function in C++ that would read contents from the file SPORTS.DAT and creates a file TENNIS.DAT copying only those records from SPORTS.DAT where the event name is "TENNIS".
5. (a) What is the difference between a tuple and an attribute in a table? Explain with a suitable example. 2
- (b) Consider the following tables Stationary and Consumer. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii). 6

Table: Stationary

S_ID	StationaryName	Company	Price
DP01	Dot Pen	ABC	10
PL02	Pencil	XYZ	6
ER05	Eraser	XYZ	7
PL01	Pencil	CAM	5
GP02	Gel Pen	ABC	15

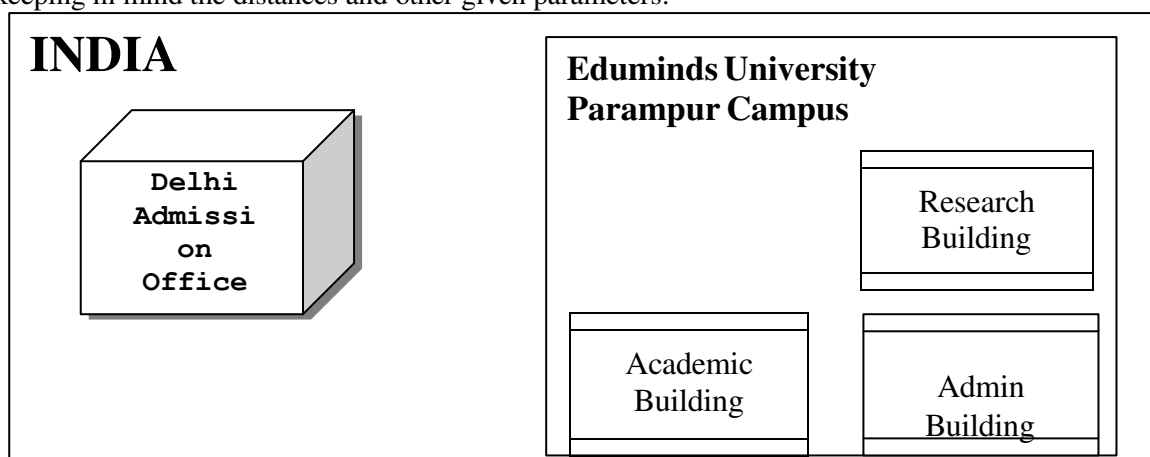
Table: Consumer

C_ID	ConsumerName	Address	S_ID
01	Good Learner	Delhi	PL01
06	Write Well	Mumbai	GP02
12	Topper	Delhi	DP01
15	Write And Draw	Delhi	PL02
16	Motivation	Bangalore	PL01

- (i) To display the details of those Consumers whose Address is Delhi.
(ii) To display the details of Stationary whose Price is in range of 8 to 15 (both values included).
(iii) To display the ConsumerName, Address from table Consumer and Company and Price from table Stationary with their corresponding matching S_ID.
(iv) To increase the Price of all stationary by 2.
(v) SELECT DISTINCT Address FROM Consumer;
(vi) SELECT Company, MAX(Price), MIN(Price), Count(*) FROM Stationary GROUP BY Company;
(vii) SELECT Consumer.ConsumerName, Stationary.StationaryName, Stationary.Price FROM Stationary, Consumer WHERE Consumer.S_ID = Stationary.S_ID;
(viii) SELECT StationaryName , Price * 3 FROM Stationary;
6. (a) State and verify Distributive Law in Boolean Algebra. 2
(b) Write the dual of the Boolean expression $A + B \cdot C$ 1
(c) Realise $XY \cdot Z + X \cdot Y \cdot Z$ using NAND gates only. 2
(d) Reduce the following Boolean Expression using K-map: 3

$$F(P,Q,R,S) = \sum (0, 1, 3, 4, 5, 6, 7, 9,10,11,13,15)$$
7. (a) Differentiate between uploading and downloading. 1
(b) Expand the following terms: 2
(i) HTTP
(ii) SLIP
(iii) SMS
(iv) W3C
(c) What is a protocol? Name any two? 1
(d) What are the following softwares used for? 2
(i) Mozilla
(ii) Linux
(e) "Eduminds University" is starting its first campus in a small town Parampur of Central India with its centre admission office in Delhi. The University has 3 major buildings comprising of Admin building, Academic Building and Research Building in the 5 KM area Campus. 4

As a network expert, you need to suggest the network plan as per (i) to (iv) to the authorities keeping in mind the distances and other given parameters.



Expected wire distances between various locations:

Research Building to Admin Building	90m
Research Building to Academic Building	80m

Academic Building to Admin Building	15m
Delhi Admission Office to Parampur campus	1450Km

Expected number of computers to be installed at various locations in the University are as follows:

Research Building	20
Academic Building	150
Admin Building	35
Delhi Admission Building	5

- (i) Suggest to the authorities, the cable layout amongst various buildings inside University campus for connecting the buildings.
- (ii) Suggest the most suitable place (i.e. building) to house the server of this organization with a suitable reason.
- (iii) Suggest an effective device from the following to be installed in each of the buildings to connect all the computers:
 - Gateway
 - Modem
 - Switch
- (iv) Suggest the most suitable (very high speed) service to provide data connectivity between Admission building located in Delhi and the campus located in Parampur from the following options:
 - Telephone line
 - Fixed line Dial-up connection
 - Co-axial Cable Network
 - GSM
 - Satellite Connection

Marking Scheme

Important Note:

- ❖ The answers given in the marking scheme are SUGGESTIVE. Examiners are requested to award marks for all alternative correct solution/answers conveying the similar meaning.

EXPECTED ANSWER

1. (a) typedef defines an alias name for an existing type.

Example: typedef float Amount;
Amount principle, loan, balance;

(1 mark for purpose)

(1 mark for suitable example)

- (b) (iii) math.h
(iv) conio.h

(1/2 marks for each correct header file)

- (c)

```
#include<iostream.h>
#include<stdio.h>
class Candidate
{
    int CandidateId;
    char Name[20];
public:
```

```

Candidate() { }
void Register() {cin>>CandidateId; gets(Name);}
void List() {cout<<CandidateId<<": "<<Name<<endl;}
};
void main()
{
Candidate C;
C.Register();
C.List();
}

```

(1/2 marks for each error correction)

- (d) 78#80#83#
234@240@249@249@
(1½ marks for each output line)

- (e) QRfApAsD
(2 marks correct output)

- (f) Maximum value: 41
Minimum value: 70
(1 mark for each value)

2. (a) A constructor is a member function having the same name as that of the class and which gets invoked every time a new object is created. It is used to construct and initialize object values. A destructor function has the same name as that of constructor function, preceded with a ~-(tilde) sign. It gets invoked every time an object goes out of scope. It is used to destroy objects.

Example:

```

class A
{ int a, b;
public:
A() //Constructor
{a=5; b=10;}
~A() //Destructor
{cout<<"Destructor invoked"};
.....
};

```

(1/2 mark for each definition)

(1 mark for suitable example)

- (b) (i) No. Since the function 1 (Product()) being default constructor is defined inside private section, it cannot initialize the objects declared outside the class.

Correction needed is that the constructor Product() should be declared inside the public section.

(ii) Computer-Notebook:20@35000

(1 mark for each correct answer)

- (c) *(1 mark for private data member declaration)*
(1 mark for each function definition)
(1/2 marks should be deduced if public member functions are defined inside the class)

- (d) (v) Hierarchical Inheritance
(vi) 50 bytes
(vii) match_fee, BOWLERName, Wickets
(viii) CInput(), CShow(), BATInput(), BATShow()
(1 mark for each answer)

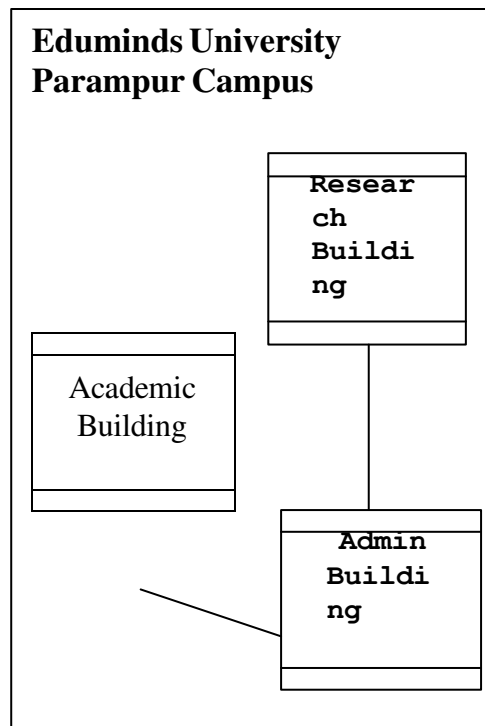
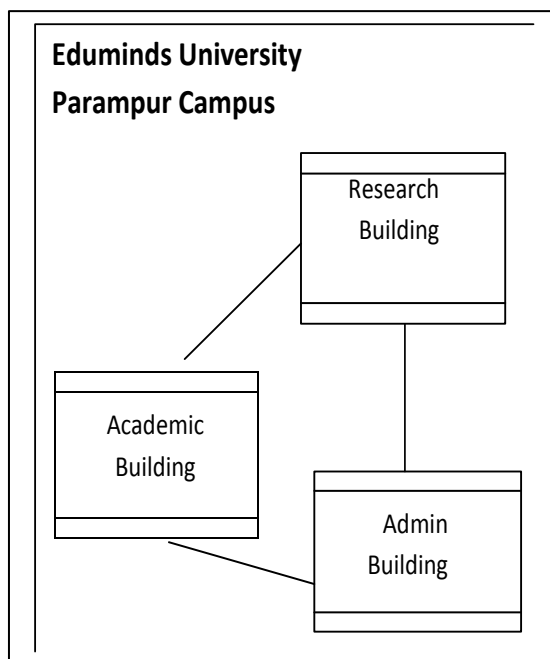
3. (a) (1/2 mark for function prototype)
(2½ marks for sorting using insertion sort)
(No mark will be given if sorting is done by using any other sorting technique)
- (b) Base address : 7280
Address of Array[30][10] : 9000
(1½ mark for base address calculation)
(1½ mark for Array[30][10] address calculation)
- (c) (1mark for correct logic)
(2 marks for function definition)
- (d) (1½ 1mark for each sum)
- (e) **A B C D - * E / +**
(1mark for showing stack content)
(1mark for correct answer)
4. (a) The ios::out is used to overwrite the existing data file which creates the file newly and the ios::app is used to append the data in the existing file.
(½ marks for each mode)
- (b) (½ marks for opening the file)
(1½ marks for correct function definition)
- (c) (½ marks each for opening and closing file)
(1 mark each for reading data from file and writing data in file)
5. (a) A row in a relation is called a tuple.
A column in a relation is called an attribute.
(1/2 marks for each definition)
(1 mark for suitable example)
- (b) (ix) SELECT * FROM Consumer WHERE Address = „Delhi“;
(x) SELECT * FROM Stationary WHERE Price BETWEEN 8 AND 15;
(xi) SELECT C.ConsumerName, C.Address, S.Company, S.Price FROM Consumer C, Stationary S WHERE C.S_ID=S.S_ID;
(xii) UPDATE Stationary SET Price=Price+ 2;
(xiii) Address
Delhi
Mumbai
Bangalore
- (xiv)

<u>Company</u>	<u>MAX(Price)</u>	<u>MIN(Price)</u>	<u>Count(*)</u>
ABC	15	10	2
XYZ	7	6	2
CAM	5	5	1
- (xv)

<u>ConsumerName</u>	<u>StationaryName</u>	<u>Price</u>
Good Learner	Pencil	5
Write Well	Gel pen	15
Topper	Dot Pen	10
Write And Draw	Pencil	6
Motivation	Pencil	5
- (xvi)

<u>StationaryName</u>	<u>Price * 3</u>
Dot Pen	30
Pencil	18
Eraser	21
Pencil	15
Gel Pen	45
- (1 mark for each query (i) to (iv))
(½ marks for each output (v) to (viii))
6. (a) (i) $A + B.C = (A+B).(A+C)$

- (ii) $A.(B+C) = A.B + A.C$
(½ marks for each statement)
(½ marks for each validation)
- (b) $A . B''+C$
(1 mark for correct answer)
- (c) *(2 marks for correct circuit diagram)*
(no mark should be given if used any other gate)
- (d) $F(P,Q,R,S) = S + P''R'' + P''Q + PQ''R$
(1 mark for correct K-map)
(2 marks for correct answer)
7. (a) Uploading means transferring a file from your computer to your home directory on the host system and downloading means transferring a file from a remote computer to your computer.
(½ mark for each definition)
- (b) (v) Hyper Text Transfer Protocol
 (vi) Serial Line Internet Protocol
 (vii) Short Message Service
 (viii) World Wide Web Consortium
(½ marks for each answer)
- (c) Protocol is a set of rules that two or more machines must follow to exchange message.
 TCP/IP, FTP, HTTP etc are some examples of protocol.
(½ marks for definition)
(½ marks for examples)
- (d) (i) Mozilla is a free, cross-platform internet suite, whose components include a web browser, an e-mail and news client, an HTML editor.
 (ii) Linux is a popular operating system. It is a free software.
(1 marks for each answer)
- (e) (i)



OR

- (ii) Academic Building because it has the maximum number of computers.
- (iii) Switch
- (iv) Satellite Connection

(1 mark for each answer)