Class 10 Computer

Concepts • Programming Logic • Practice

A practical revision companion

Learn the concept, understand the method, practise with guided examples, and revise with confidence.

Concept Clarity • Guided Practice • Exam Confidence

01 Computer Fundamentals

Core idea

A computer accepts input, processes data using instructions, stores information and produces output. Hardware is physical; software is a set of instructions.

Key formulas / rules

Input → Processing → Storage/Memory → Output. Hardware and software work together.

Guided solved examples

Example 1. Differentiate hardware and software.

Solution. Hardware is physical equipment; software consists of instructions/programs.

Example 2. Give one example of an input device.

Solution. Keyboard.

Common mistakes

Do not call a program hardware.

Exam tip

Use examples to support definitions.

Practice

1. Give two output devices.
2. Explain the role of software.
3. Describe the basic information-processing cycle.

02 Algorithms & Flowcharts

Core idea

An algorithm is a finite sequence of unambiguous steps. A flowchart represents logic graphically.

Key formulas / rules

Common flowchart ideas: Start/End, Input/Output, Process, Decision. Algorithms should be finite and clear.

Guided solved examples

Example 1. Design logic to find the larger of two numbers.

Solution. Read A,B; if A>B output A; otherwise output B.

Example 2. What does a decision symbol represent?

Solution. A condition whose outcome determines which path is followed.

Common mistakes

Separate input, processing and output steps.

Exam tip

Trace the algorithm manually with sample values.

Practice

1. Write an algorithm to find the average of three numbers.
2. Create logic to check even/odd.
3. List four qualities of a good algorithm.

03 Programming Logic

Core idea

Programs use sequence, selection and iteration to express algorithms. Variables store values that change during execution.

Key formulas / rules

Sequence: order; Selection: if/else choice; Iteration: repeat using a condition or count.

Guided solved examples

Example 1. What structure chooses between two alternatives?

Solution. Selection, commonly represented by if/else logic.

Example 2. Why use a loop?

Solution. To repeat a set of instructions efficiently when a condition or count requires repetition.

Common mistakes

Check loop start, stop and update conditions.

Exam tip

Dry-run the program using a small table of variable values.

Practice

1. Write pseudocode for checking positive/negative.
2. Write pseudocode to sum numbers 1 to 10.
3. Explain sequence vs iteration.

04 Data & Debugging

Core idea

Data types describe the kind of value a variable stores. Debugging is the systematic process of finding and correcting program errors.

Key formulas / rules

Common errors: syntax, runtime and logical. Use meaningful variable names and test boundary cases.

Guided solved examples

Example 1. A program runs but gives the wrong result. What type of error may this be?

Solution. A logical error is a likely cause.

Example 2. Why test boundary values?

Solution. They can expose errors that normal inputs do not reveal.

Common mistakes
Don't assume the first output is correct; test multiple cases.

Exam tip
Record the input, expected output and actual output during debugging.

Practice

1. Give one syntax-error example.
2. Give one logical-error example.
3. Design three test cases for a marks calculator.

05 Problem Solving

Core idea

Good programming begins before coding: understand the problem, define input/output, design logic, test and refine.

Key formulas / rules
Workflow: Understand → Break down → Pseudocode → Code → Test → Debug → Improve.

Guided solved examples

Example 1. Why write pseudocode first?

Solution. It separates problem logic from programming-language syntax and makes errors easier to spot.

Example 2. What makes a good test case?

Solution. It has a known expected result and covers a normal, boundary or unusual condition.

Common mistakes
Avoid coding before understanding the required output.

Exam tip
Use small test cases first, then more complex cases.

Practice

1. Create a flowchart idea for grading marks.
2. Design test cases for an even/odd program.
3. Explain why edge cases matter.

Rapid Revision Sheet

Core Cycle
Input → Processing → Storage → Output.
Algorithms
Finite, clear and ordered steps; define inputs and expected outputs.
Programming
Sequence, selection and iteration are the core control structures.
Debugging
Check syntax, runtime behaviour and logic; test normal and boundary cases.
Problem Solving
Understand → Pseudocode → Code → Test → Debug.

Mixed Practice Test

1. Write pseudocode to find the largest of three numbers.
2. Explain hardware vs software.
3. Name the three core control structures.
4. Give one logical-error example.
5. Design two test cases for an even/odd program.
6. Why is pseudocode useful?

Answer Key

1. Read A,B,C; compare values; output the largest.
2. Hardware is physical; software is instructions/programs.
3. Sequence, selection, iteration.
4. Any valid example where the program runs but computes the wrong result.
5. For example 8→even and 7→odd.
6. It clarifies logic before language-specific coding.

Ready to Learn with Clarity?

Don’t just memorise formulas — understand the concepts, practise the right approach and build confidence.

Book Your Demo Class

Book a demo class today and experience a structured, concept-focused teaching approach tailored to your learning needs.

Start learning smarter. Book your demo class today.

Mathematics • Physics • Computer • Personalised Guidance