ISC Class 12 Computer Science (with Java) (CS)

Unit-wise syllabus and marks for the 2026-27 session.

UNITS	SESSION
13	2026-27

Exam pattern. Total 100 marks. Paper 1 (Theory) 70 marks, 3 hours: Part I (20 marks) compulsory short-answer questions covering the whole syllabus; Part II (50 marks) with Section A and Section B (10 marks each) and Section C (5 marks), candidates answering set numbers of questions. Paper 2 (Practical) 30 marks, 3 hours: candidates write a working Java program for one of three given problems, evaluated by a visiting examiner.

Practical. Paper 2 is a 3-hour, 30-mark practical conducted in BlueJ/JDK where students design, code and test a Java program for one of three problems (assessed on planning, coding, output and viva). Programs are documented in a practical file/lab record maintained through the year, reflecting algorithm design, data structures and OOP work.

#	UNIT AND TOPICS	MARKS
1	**Boolean Algebra** Propositional logic, well-formed formulae, truth tables and interpretation; Satisfiable, unsatisfiable and valid formulae; Binary-valued quantities; AND, OR, NOT operations and equivalence laws; Basic theorems: duality, idempotence, commutativity, associativity, distributivity; De Morgan's laws and applications; Canonical SOP and POS forms; minimization; Karnaugh maps for minimization of Boolean expressions (up to 4 variables)	–
2	**Computer Hardware** Elementary logic gates: NOT, AND, OR, NAND, NOR, XOR, XNOR; Universal gates (NAND, NOR) and circuit applications; Half adder and full adder; Encoders, decoders and multiplexers using Boolean algebra	–
3	**Implementation (Algorithms to Programs)** From algorithms to working Java programs; Programming environment (BlueJ, JDK); Review of Class XI Java concepts	–
4	**Objects and Classes** Objects as attributes (state) and behaviour (methods); Object as an instance of a class; Constructors: default, parameterized and copy constructors; Static members and static methods; Encapsulation and access specifiers	–
5	**Primitive Values, Wrapper Classes, Types and Casting**	–

#	UNIT AND TOPICS	MARKS
	Primitive types: int, short, long, float, double, boolean, char; Wrapper classes and autoboxing/unboxing; User-defined types via classes; Type conversion, coercion and casting	
6	**Variables and Expressions** Variables as named values; Arithmetic, relational and logical expressions; Operators, associativity and precedence; Assignment operations	–
7	**Statements and Scope** Conditional statements: if, if-else, switch, ternary operator; Loops: for, while, do-while; break and continue; Blocks, scope and variable visibility	–
8	**Functions (Methods)** Methods as abstractions of operations; side effects; Formal vs actual parameters; pass-by-value vs object references; Static methods and the 'this' reference; Algorithmic problem solving with methods	–
9	**Arrays and Strings** Single and multi-dimensional arrays; row-major/column-major addressing; String class methods and manipulation; Searching: linear and binary search; Sorting: bubble, selection, insertion sort; Exception handling (try, catch, throw); Basic file I/O and the Scanner class; tokenisation	–
10	**Recursion** Recursive definitions and base cases; Factorial, GCD, Fibonacci, binary search; Conversion of number representations; Recursive sorting techniques	–
11	**Inheritance, Interfaces and Polymorphism** Single, multilevel, hierarchical and hybrid inheritance; Base and derived classes; super and protected access; Abstract classes and the Object class; Interfaces and multiple inheritance of type; Method overloading and overriding; dynamic (run-time) polymorphism	–
12	**Data Structures** Stack, queue and dequeue implementations; Interfaces and multiple implementations; Infix to prefix and postfix conversion and evaluation; Single linked lists: insertion, deletion, traversal; Binary trees: root, nodes, leaves, height, depth, level, size, degree, siblings, subtree, completeness, balancing; Tree traversals: pre-order, in-order, post-order (conceptual)	–
13	**Computational Complexity and Big O Notation** Input size and estimating computational complexity; Dominant term and order of growth; Best, average and worst case analysis; Big O notation and analysing algorithm complexity	–