ISC

INDIAN SCHOOL CERTIFICATE EXAMINATION

YEAR 2028

COMPUTER SCIENCE

(868)

Developed by:
Research, Development and Curriculum Division (RDCD)
CISCE

January 2026

Council for the Indian School Certificate Examinations (CISCE)

MISSION STATEMENT

The Council for the Indian School Certificate Examinations is committed to serving the nation's children, through high quality educational endeavours, empowering them to contribute towards a humane, just and pluralistic society, promoting introspective living, by creating exciting learning opportunities, with a commitment to excellence.

ETHOS OF CISCE

- Trust and fair play.
- Minimum monitoring.
- Allowing schools to evolve their own niche.
- Catering to the needs of the children.
- Giving freedom to experiment with new ideas and practices.
- Diversity and plurality - the basic strength for evolution of ideas.
- Schools to motivate pupils towards the cultivation of:
  Excellence - The Indian and Global experience.
  Values - Spiritual and cultural - to be the bedrock of the educational experience.
- Schools to have an 'Indian Ethos', strong roots in the national psyche and be sensitive to national aspirations.

COMPUTER SCIENCE (868)

Aims (Conceptual)

(1) To understand algorithmic problem solving using data abstractions, functional and procedural abstractions, and object based and object-oriented abstractions.

(2) To understand:

  (a) how computers represent, store and process data at different levels of abstraction that mediate between the machine and the algorithmic problem solving level and

  (b) how they communicate with the outside world.

(3) To create awareness of ethical issues related to computing and to promote safe, ethical behavior.

(4) To make students aware of future trends in computing.

Aims (Skills)

To devise algorithmic solutions to problems and to be able to code, validate, document, execute and debug the solution using the Java and python programming system to align with the future trend.

CLASS XI

*There will be **two** papers in the subject:*

***Paper I:** Theory…….* *3 hours…70 marks*

***Paper II:** Practical…….* *3 hours…30 marks*

Distribution of marks for the theory paper:

S.no	Unit	Marks
	Section A – Hardware	
1.	System of Numeration	25
2.	Encodings	
3.	Propositional logic, Hardware implementation, Arithmetic operations	
	Section B – Java	
4.	Introduction to Object Oriented Programming using Java	30
5.	Objects	
6.	Primitive values, Wrapper classes, Types and casting	
7.	Variables, Expressions	
8.	Statements, Scope	
9.	Methods and Constructors	
10.	Arrays, Strings	
11.	Basic input/ output file handling(only Text)	
	Section C – Python and Conceptual aspects	
12.	Introduction to python	15
13.	Trends in computing and ethical issues	
	Total	**70**

PAPER I (THEORY) : 70 MARKS

SECTION A - HARDWARE

1. System of Numeration

Representation of numbers in different bases and interconversion between them (e.g. binary, octal, decimal, hexadecimal). Addition, subtraction and multiplication operations for numbers in different bases.

Introduce the positional system of representing numbers and the concept of a base. Discuss the conversion of representations between different bases using English or pseudo code. These algorithms are also good examples for defining different functions in a class modelling numbers (when programming is discussed). For addition and subtraction (1's complement and 2's complement) use the analogy with decimal numbers, emphasize how carry works (this will be useful later when binary adders are discussed), multiplication of numbers in different bases [without decimals].

2. Encodings

Characters and their encodings (e.g. ASCII, ISCII, Unicode).

Discuss the limitations of the ASCII code in representing characters of other languages. Discuss the Unicode representation for the local language. Java uses Unicode, so strings in the local language can be used (they can be displayed if fonts are available) – a simple table lookup for local language equivalents for Latin (i.e. English) character strings may be done. More details on Unicode are available at www.unicode.org.

3. Propositional logic, Hardware implementation, Arithmetic operations

(a) Propositional logic, well-formed formulae, truth values and interpretation of well formed formulae, truth tables.

*Propositional variables; the common logical connectives ~ (negation) ∧ (and)(conjunction),∨ (or)(disjunction), ⇒ (implication), ⇔ (equivalence)); definition of a well-formed formula (wff); representation of simple word problems as wff (this can be used for motivation); the values **true** and **false**; interpretation of a wff; truth tables; satisfiable, unsatisfiable and valid formulae.*

(b) Logic and hardware, basic gates (AND, NOT, OR) and their universality, other gates (NAND, NOR, XOR, XNOR), half adder, full adder.

Show how the logic in (a) above can be realized in hardware in the form of gates. These gates can then be combined to implement the basic operations for arithmetic. Tie up with the arithmetic operations on integers discussed earlier in 2.

SECTION B - JAVA

The programming element in the syllabus (Section B and C) is aimed at algorithmic problem solving and **not** merely rote learning of Java and Python syntax. The Java version used should be 6.0 or later and Python version 3.13 and above.

For Java programming, students may use any text editor along with the javac compiler, or any integrated development environment (IDE) such as **BlueJ**, **DrJava**, **Eclipse**, or **NetBeans**.

For Python programming, students may use any suitable development environment, such as **IDLE**, **PyCharm**, **VS Code**, **Spyder**, or **Jupyter Notebook**.

The students are required to do lab assignments in the computer lab concurrently with the lectures. Programming assignments should start with development of appropriate algorithm followed by coding using Java and Python.

4. Introduction to Object Oriented Programming using Java

Note that topics 5 to 11 should be introduced almost simultaneously along with Classes and their definitions.

5. Objects

(a) Objects as data (attributes) + behaviour (methods or methods); object as an instance of a class.

Difference between object and class should be made very clear. BlueJ (www.bluej.org) and Greenfoot (www.greenfoot.org) can be used for this purpose.

(b) Analysis of some real-world programming examples in terms of objects and classes.

Use simple examples like a calculator, date, number etc. to illustrate how they can be treated as objects that behave in certain well- defined ways and how the interface provides a way to access behaviour. Illustrate behaviour changes by adding new methods, deleting old methods or modifying existing methods.

(c) Basic concept of a virtual machine; Java Virtual Machine (JVM); compilation and execution of Java programs (the javac and java programs).

The JVM is a machine but built as a program and not through hardware. Therefore it is called a virtual machine. To run, JVM machine language programs require an interpreter. The advantage is that such JVM machine language programs (.class files) are portable and can run on any machine that has the java program.

(d) Compile time and run time errors; basic concept of an exception, the Exception class, try-catch, throw, throws and finally.

Differentiate between compile time and run time errors. Run time errors crash the program. Recovery is possible by the use of exceptions. Explain how an exception object is created and passed up until a matching catch is found. This behaviour is different from the one where a value is returned by a deeply nested method call.

6. Primitive values, Wrapper classes, Types and casting

Primitive values and types: byte, int, short, long, float, double, boolean, char. Corresponding wrapper classes for each primitive type. Class as type of the object. Class as mechanism for user defined types. Changing types through user defined casting and automatic type coercion for some primitive types.

Ideally, everything should be a class; primitive types are defined for efficiency reasons; each primitive type has a corresponding wrapper class. Classes as user defined types. In some cases, types are changed by automatic coercion or casting – e.g. mixed type expressions. However, casting in general is not a good idea and should be avoided, if possible.

7. Variables, Expressions

Variables as names for values; named constants (final), expressions (arithmetic and logical) and their evaluation (operators, associativity, precedence). Assignment operation; difference between left-hand side and right-hand side of assignment.

Variables denote values; variables are already defined as attributes in classes; variables have types that constrain the values it can denote. Difference between variables denoting primitive values and object values – variables denoting objects are references to those objects.

NOTE: Library functions for solving expressions may be used as and when required.

8. Statements, Scope

Statements; conditional (if, if else, if else if, switch case) ternary operator, looping (for, while, do while), continue, break; grouping statements in blocks, scope and visibility of variables.

Describe the semantics of the conditional and looping statements in detail. Evaluation of the condition in conditional statements.
Nesting of blocks. Variables with block scope, method scope, class scope. Visibility rules when variables with the same name are defined in different scopes.

9. Methods and Constructors

Methods and Constructors (as abstractions for complex user defined operations on objects), methods as mechanisms for side effects; formal arguments and actual arguments in methods; different behaviour of primitive arguments. Static methods and variables. The ***this*** operator. Examples of algorithmic problem solving using methods (number problems, finding roots of algebraic equations etc.).

*Methods are like complex operations where the object is implicitly the first argument. Operator **this** denotes the current object. Methods typically return values. Static definitions as class variables and class methods are visible and shared by all instances. Need for static methods and variables. Introduce the main method – needed to begin execution. Constructor as a special kind of method; the new operator; multiple constructors with different argument structures; constructor returns a reference to the object.*

10. Arrays, Strings

Structured data types – arrays (single and multi- dimensional), Strings. Example algorithms that use structured data types (searching, finding maximum/minimum, sorting techniques, solving systems of linear equations, substring, concatenation, length, access to char in string, etc.).

Storing many data elements of the same type requires structured data types – like arrays. Access in arrays is constant time and does not depend on the number of elements. Sorting techniques (bubble, selection, insertion), Structured data types can be defined by classes – String. Introduce the Java library String class and the basic operations on strings (accessing individual characters, various substring operations, concatenation, replacement, index of operations).

11. Basic input/output and Text File Handling

(a) Basic input/output using Scanner class.

Input/output exceptions. Tokens in an input stream, concept of whitespace, extracting tokens from an input stream (String Tokenizer class). The Scanner class can be used for input of various types of data (e.g. int, float, char etc.) from the standard input stream.

Discuss the concept of a token (a delimited continuous stream of characters that is meaningful in the application program – e.g. words in a sentence where the delimiter is the blank character). This naturally leads to the idea of delimiters and in particular whitespace and user defined characters as delimiters. As an example show how the String Tokenizer class allows one to extract a sequence of tokens from a string with user defined delimiters.

(b) Data File Handling.

Need for Data file, Input Stream, Output Stream, Character Stream (FileReader, FileWriter), Operations- Creation, Reading, Writing, Appending, and Searching.

SECTION C - PYTHON AND CONCEPTUAL ASPECTS

12. Introduction to Python

(a) Installation and IDE

Source : www.python.org
IDE : IDLE, Pycharm, VScode, spyder, Jupyter

(b) Fundamentals of Python programming

Easy to use and learn, simple syntax, Open Source, Large Standard Library, Dynamically Typed, Large Community Support, Portable, Platform Independent, Interpreter based.
Execution modes - interactive mode and script mode, Python character set, tokens (keyword, identifier, literal, operator, punctuator), variables, concept of l-value and r-value, use of comments.

(c) Data types

Number(integer, floating point, complex), boolean, sequence(string, list, tuple), None, Mapping(dictionary), mutable and immutable data types.

(d) Data processing in Python

Accepting data as input from the console and displaying output. Errors- syntax errors, logical errors, and run-time errors

(e) Operators and expressions

Forms of operators, Expressions, Statements and Type conversions

*Forms of operators (Unary, binary, ternary) , Arithmetic operators (+, - , *, **, /, //,%) Relational operators (> , <, >=, <=, ==, !=), Logical operators(and, or, not), Assignment operators (=) , Augmented assignment operators(+=, -=, *=, /=,%=,//=, **=), Identity operators (is, is not), membership operators (in, not in), Python expressions/statement, evaluation of the expressions involving the operators, type conversion, precedence of operators, type-conversion (explicit and implicit conversion)*

(f) Flow of control

Introduction of programming constructs, use of indentation, sequential flow, conditional and iterative flow

Conditional statements (if, if-else, if-elif-else), Iterative Statement(for loop, range(), while loop), break and continue statements, nested loop.

13. Trends in computing and ethical issues

(a) Artificial Intelligence, Internet of Things, Virtual Reality and Augmented Reality.

(b) Cyber Security, privacy, netiquette, spam, phishing, Digital arrest.

(c) Intellectual property, Software copyright and patents and Free Software Foundation.

Intellectual property and corresponding laws and rights, software as intellectual property.

Software copyright and patents and the difference between the two; trademarks; software licensing and piracy. free Software Foundation and its position on software, Open Source Software, various types of licensing (e.g. GPL, BSD).

Brief understanding of the above , the social impact and ethical issues should be discussed and debated in class. The important thing is for students to realise that these are complex issues and there are multiple points of view on many of them and there is no single 'correct' or 'right' view.

PAPER II (PRACTICAL) : 30 MARKS

This paper of three hours duration will be evaluated internally by the school.
The paper shall consist of three programming problems (in Java only) from which a candidate has to attempt any one. The practical consists of the two parts:

(1) Planning Session
(2) Examination Session

The total time to be spent on the Planning session and the Examination session is three hours. A maximum of 90 minutes is permitted for the Planning session and 90 minutes for the Examination session. **Candidates are to be permitted to proceed to the Examination Session only after the 90 minutes of the Planning Session are over.**

Planning Session

The candidates will be required to prepare an algorithm and a hand-written Java program to solve the problem.

Examination Session

The program handed in at the end of the Planning session shall be returned to the candidates. The candidates will be required to key-in and execute the Java program on seen and unseen inputs individually on the Computer and show execution to the examiner. A printout of the program listing, including output results should be attached to the answer script containing the algorithm and handwritten program. This should be returned to the examiner. The program should be sufficiently documented so that the algorithm, representation and development process is clear from reading the program. Large differences between the planned program and the printout will result in loss of marks.Teachers should maintain a record of all the assignments done as part of the practical work throughout the year and give it due credit at the time of cumulative evaluation at the end of the year. Students are expected to do a **minimum** of twenty assignments (15 programs in Java and 5 programs in Python) for the year and ONE project based on the chapter 'Trends in computing and ethical issues'. The project work may be submitted to the school in handwritten or typed form (hard copy or soft copy), or as PowerPoint presentations (PPTs).

TOPIC WISE BREAKUP OF ASSIGNMENTS

Java – minimum of 15 programs

- Three programs based of conditional / iterative statements (series, number logic, patterns)
- Three programs based on Single dimensional arrays
- Three programs based on Double dimensional arrays
- Four programs based on String handling
- Two programs based on File handling

Note: All programs should be based on classes and objects.

Python – minimum of 5 programs

- One program based on sequential statements
- Two programs on conditional / iteration statements
- Two programs on nested iteration statements

LIST OF SUGGESTED PROJECTS:

PRESENTATION / MODEL BASED/ APPLICATION BASED

1. Creating an expert system for road-traffic management (routing and re-routing of vehicles depending on congestion).
2. Creating an expert system for medical diagnosis on the basis of symptoms and prescribe a suitable treatment.
3. Creating a security system for age-appropriate access to social media.

4. Simulate Adders using Arduino Controllers and Components.
5. Simulate a converter of Binary to Decimal number systems using Arduino Controllers and Components.
6. Develop a console-based application using Java for Flight Ticket Reservation.
7. Develop a console-based application using Java to encrypt and decrypt a message (using cipher text, Unicode-exchange, etc).
8. Develop a console-based application using Java to find name of the bank and branch location from IFSC.
9. Develop a console-based application using Java to calculate taxable income (only direct tax).
10. Develop a console-based application using Java to develop a simple text editor (text typing, copy, cut, paste, delete).

EVALUATION

Marks (out of a total of 30) should be distributed as given below:

Continuous Evaluation

Candidates will be required to submit a work file containing the practical work related to programming assignments done during the year and **ONE** project.

Programming assignments done throughout the year	10 marks
Project Work - (based on any topic from the syllabus)	5 marks

Terminal Evaluation

Solution to Java programming problem on the computer	15 Marks

Marks should be given for choice of algorithm and implementation strategy, documentation, correct output on known inputs mentioned in the question paper, correct output for unknown inputs available only to the examiner.

CLASS XII

*There will be **two** papers in the subject:*
***Paper I:** Theory……..3 hours….70 marks*
***Paper II:** Practical……3 hours….30 marks*

Distribution of marks for the theory paper:

S.No.	Topic	Marks
	Section A – Hardware	
1.	Boolean Algebra	25 Marks
2.	Computer Hardware	
	Section B - Java	
3.	Methods – Object as a parameter and return data type	30 Marks
4.	Arrays (Single and double dimensional), String handling	
5.	Recursion	
6.	Inheritance and Interface	
	Section C - Python & Conceptual aspects	
7.	Programming in Python	15 Marks
8.	Data Structures	
9.	Complexity and Big O notation	
	Total	70 Marks

PAPER I (THEORY) : 70 MARKS

SECTION A - HARDWARE

1. Boolean Algebra

(a) Propositional logic, well formed formulae, truth values and interpretation of well formed formulae (wff), truth tables, satisfiable, unsatisfiable and valid formulae. Equivalence laws and their use in simplifying wffs.

Propositional variables; the common logical connectives (~ (not)(negation), ∧ (and)(conjunction), ∨ (or)(disjunction), ⇒ (implication), ⇔ (biconditional); definition of a well-formed formula (wff);
*`representation of simple word problems as wff (this can be used for motivation); the values **true** and **false**; interpretation of a wff; truth tables; satisfiable, unsatisfiable and valid formulae.*

*Equivalence laws: commutativity of ∧, ∨; associativity of ∧, ∨; distributivity; De Morgan's laws; law of implication ($p \Rightarrow q \equiv \sim p \vee q$); law of biconditional ($(p \Leftrightarrow q) \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$); identity ($p \equiv p$); law of negation ($\sim(\sim p) \equiv p$); law of excluded middle ($p \vee \sim p \equiv$ **true**); law of contradiction ($p \wedge \sim p \equiv$ **false**); tautology and contingency simplification rules for ∧, ∨. Converse, inverse and contra positive. Chain rule, Modus ponens.*

(b) Binary valued quantities; basic postulates of Boolean algebra; operations AND, OR and NOT; truth tables.

(c) Basic theorems of Boolean algebra (e.g. duality, idempotence, commutativity, associativity, distributivity, operations with 0 and 1, complements, absorption, involution); De Morgan's theorem and its applications; reducing Boolean expressions to sum of products and product of sums forms; Karnaugh maps (up to four variables).

Verify the laws of Boolean algebra using truth tables. Inputs, outputs for circuits like half and full adders, majority circuit etc., SOP and POS representation; Maxterms & Minterms, Canonical and Cardinal representation, reduction using Karnaugh maps and Boolean algebra.

2. Computer Hardware

(a) Elementary logic gates (NOT, AND, OR, NAND, NOR, XOR, XNOR) and their use, performance, expression, truth table and symbol.

(b) Applications of Boolean algebra and logic gates to half adders, full adders, encoders, decoders, multiplexers, half subtractors, full subtractors, NAND, NOR as universal gates.

Show the correspondence between Boolean methods and the corresponding switching circuits or gates. Show that NAND and NOR gates are universal by converting some circuits to purely NAND or NOR gates.

SECTION B - JAVA

The programming element in the syllabus (Sections B and C) is aimed at algorithmic problem solving and **not** merely rote learning of Java and Python syntax. The Java version used should be 6.0 or later and Python version 3.13 and above.

Recapitulation of Class XI Sections B - Programming in Java

(i) Introduction to object oriented programming using Java (ii) Objects (iii) Primitive values, Wrapper classes, Types and casting (iv) Variables, Expressions (v) Statements, Scope (vi) Methods and constructors (vii) Arrays ,Strings

While recapitulating, ensure that higher order problems are solved using these constructs.

3. Methods - Object as parameter and return data type

Methods (as abstractions for complex user defined operations on objects), formal arguments and actual arguments in methods; different behaviour of primitive and object arguments. Static method and variables. The **this** Operator – as a reference of the current object. Passing objects /arrays as parameters to a method and returning an object / array.

4. Arrays (Single and Double dimensional), String handling

(a) Structured data types – arrays (single and multi- dimensional), address calculations. Algorithms that use structured data types (e.g. searching, finding maximum/minimum, sorting techniques, solving systems of linear equations)

Storing many data elements of the same type requires structured data types – like arrays. Access in arrays is constant time and does not depend on the number of elements. Address calculation (row major and column major), Sorting techniques (bubble, selection, insertion).

(b) String handling - substring, concatenation, length, access to char in string, etc.

Structured data types can be defined by classes – String. Introduce the Java library String class and the basic operations on strings (accessing individual characters, various substring operations, concatenation, replacement, index of operations). The class StringBuffer should be introduced for those applications that involve heavy manipulation of strings.

5. Recursion

Concept of recursion, simple recursive methods (e.g. factorial, GCD, binary search, conversion of representations of numbers between different bases).

Many problems can be solved very elegantly by observing that the solution can be composed of solutions to 'smaller' versions of the same problem with the base version having a known simple solution. Recursion can be initially motivated by using recursive equations to define certain methods. These definitions are fairly obvious and are easy to understand. The definitions can be directly converted to a program. Emphasize that any recursion must have a base case. Otherwise, the computation can go into an infinite loop.

The tower of Hanoi is a very good example of how recursion gives a very simple and elegant solution where as non-recursive solutions are quite complex.

6. Inheritance and Interfaces

(a) Inheritance; super and derived classes; member access in derived classes; redefinition of variables and methods in subclasses; abstract classes; class Object; protected visibility. Subclass polymorphism and dynamic binding.

Emphasize inheritance as a mechanism to reuse a class by extending it. Inheritance should not normally be used just to reuse some methods defined in a class but only when there is a genuine specialization (or subclass) relationship between objects of the super class and that of the derived class.

(b) Interfaces in Java (Only Conceptual)

Emphasize the difference between the Java language construct interface and the word interface often used to describe the set of method prototypes of a class.

SECTION C - PYTHON AND CONCEPTUAL ASPECTS

7. Programming in Python

(a) Recapitulation of Class XI Section C

(i) Data types (ii) Data Processing in Python (iii) Operators and expressions (iv) Flow of control

While recapitulating, ensure that higher order problems are solved using these constructs.

(b) *Strings: introduction, string operations (concatenation, repetition, membership and slicing), traversing a string using loops, built-in functions/methods–len(), capitalize(), title(), lower(), upper(), count(), find(), index(), endswith(), startswith(), isalnum(), isalpha(), isdigit(), islower(), isupper(), isspace(),lstrip(), rstrip(), strip(), replace(), join(), partition(), split()*

(c) *Lists: introduction, indexing, list operations (concatenation, repetition, membership and slicing), traversing a list using loops, built-in functions/methods–len(), list(), append(), extend(), insert(), count(), index(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum(); nested lists, suggested programs: finding the maximum, minimum, mean of numeric values stored in a list; linear search on list of numbers and counting the frequency of elements in a list.*

(d) *Tuples: introduction, indexing, tuple operations (concatenation, repetition, membership and slicing); built-in functions/methods – len(), tuple(), count(), index(), sorted(), min(), max(), sum(); tuple assignment, nested tuple; suggested programs: finding the minimum, maximum, mean of values stored in a tuple; linear search on a tuple of numbers, counting the frequency of elements in a tuple.*

8. *Data structures – Binary trees [Conceptual]*

Binary trees: apart from the definition the following concepts should be covered: root, internal nodes, external nodes (leaves), height (tree, node), depth (tree, node), level, size, degree, siblings, sub tree, traversals (pre, post and in-order).

9. Complexity and Big O notation

Concrete computational complexity; concept of input size; estimating complexity in terms of methods; importance of dominant term; constants, best, average and worst case.

Big O notation for computational complexity; analysis of complexity of example algorithms using the big O notation (e.g. Various searching and sorting algorithms, algorithm for solution of linear equations, etc.).

PAPER II (PRACTICAL) : 30 MARKS

This paper of three hours' duration will be evaluated by the Visiting Examiner appointed locally and approved by CISCE.
The paper shall consist of three programming problems from which a candidate has to attempt any one. The practical consists of the two parts:

1. Planning Session
2. Examination Session

The total time to be spent on the Planning session and the Examination session is three hours. A maximum of 90 minutes is permitted for the Planning session and 90 minutes for the Examination session.

Candidates are to be permitted to proceed to the Examination Session only after the 90 minutes of the Planning Session are over.

Planning Session

The candidates will be required to prepare an algorithm and a hand written Java program to solve the problem.

Examination Session

The program handed in at the end of the Planning session shall be returned to the candidates. The candidates will be required to key-in and execute the Java program on seen and unseen inputs individually on the computer and show execution to the Visiting Examiner. A printout of the program listing including output results should be attached to the answer script containing the algorithm and handwritten program. This should be returned to the examiner. The program should be sufficiently documented so that the algorithm, representation and development process is clear from reading the program. Large differences between the planned program and the printout will result in loss of marks.

Teachers should maintain a record of all assignments completed as part of the practical work throughout the year and give them due credit during the cumulative evaluation at the end of the year. Students are expected to complete a **minimum of twenty-five assignments** (20 programs in Java and 5 programs in Python) during the year. The assignments may be submitted to the school in handwritten or typed form (hard copy or soft copy), or as PowerPoint presentations (PPTs).

TOPIC WISE BREAKUP OF ASSIGNMENTS

Java – minimum of 20 programs

- Five programs based on Methods – Object as a parameter and return data type
- Three programs based on Single dimensional arrays
- Five programs based on Double dimensional arrays
- Four programs based on String handling
- Three programs based on Inheritance

  Note: All programs should be based on classes and objects.

Python – minimum of 5 programs

- One program based on Strings

- Two programs based on the concept of Lists
- Two programs based on the concept of Tuples

EVALUATION:

Marks (out of a total of **30**) should be distributed as given below:

Continuous Evaluation

Candidates will be required to submit a work file containing the practical work related to programming assignments done during the year.

Programming assignments done throughout the year (Internal Evaluation)	10 marks
Programming assignments done throughout the year (Visiting Examiner)	5 marks

Terminal Evaluation

Solution to Java programming problem on the computer	15 Marks

Marks should be given for choice of algorithm and implementation strategy, documentation, correct output on known inputs mentioned in the question paper, correct output for unknown inputs available only to the examiner.

NOTE:

Algorithm should be expressed clearly using any standard scheme such as a pseudo code.

EQUIPMENT

There should be enough computers to provide for a teaching schedule where at least three-fourths of the time available is used for programming.

Schools should have equipment/platforms such that all the software required for practical work runs properly, i.e. it should run at acceptable speeds.

Since hardware and software evolve and change very rapidly, the schools may have to upgrade them as required.

Following are the recommended specifications as of now:

The Facilities:

- A lecture cum demonstration room with a MULTIMEDIA PROJECTOR/ an LCD and O.H.P. attached to the computer.
- A white/green board with markers should be available.
- A fully equipped Computer Laboratory that allows one computer per student.
- Local area network (LAN) with high speed internet facility.
- The computers should have a minimum of 2 GB RAM and a dual core processor. The basic requirement is that it should run the operating system, Java and Python programming system at acceptable speeds.
- Good Quality printers.

Software:

- Any suitable Operating System can be used.
- JDK 6 or later and Python 3.13 or later.
- A suitable text editor for Java and Python.

SAMPLE TABLE FOR PRACTICAL WORK

S. No.	Unique Identification Number (Unique ID) of the candidate	Assessment of Practical File		Assessment of the Practical Examination (To be evaluated by the Visiting Examiner only)				TOTAL MARKS (Total Marks are to be added and entered by the Visiting Examiner)
		Internal Evaluation 10 Marks	Visiting Examiner 5 Marks	Algorithm	Java Program with internal Documentation	Hard Copy (printout)	Output	
				3 Marks	7 Marks	2 Marks	3 Marks	30 Marks
1.								
2.								
3.								
4.								
5.								
6.								
7.								
8.								
9.								
10.								

Name of the Visiting Examiner: _____

Signature:_____

Date_____