

# 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 by studying the architecture and machine language of a simple microprocessor and the 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 problems and issues related to 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 programming system.

## CLASS XI

*There will be two papers in the subject:*

**Paper I: Theory -** 3 hours ....100 marks

**Paper II: Practical -** 3 hours ....100 marks

### PAPER I -THEORY

*Paper 1 shall be of 3 hours duration and be divided into two parts.*

**Part I (30 marks):** *This part will consist of compulsory short answer questions, testing knowledge, application and skills relating to the entire syllabus.*

**Part II (70 marks):** *This part will be divided into three Sections, A, B and C. Candidates are required to answer **three** questions out of **four** from Section A and **two** questions out of **three** in each of the Sections B and C. Each question in this part shall carry 10 marks.*

### SECTION A

#### Basic Computer Hardware and Software

##### 1. Numbers

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

##### 2. Encodings

(a) Binary encodings for integers and real numbers using a finite number of bits (sign-magnitude, twos complement, mantissa-exponent notation). Basic operations on integers and floating point numbers. Limitations of finite representations.

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

##### 3. High level structure of computer

Block diagram of a computer system with details of (i) function of each block and (ii) interconnectivity and data and control flow between the various blocks.

##### 4. Basic architecture of typical simple processor and its assembly language

(a) Basic architecture of the 8085 microprocessor. Instruction set, addresses, addressing modes, simple machine language programs using the different addressing modes, execution of machine language programs, input and output.

(b) Assembly language of 8085, simple assembly language programs, assembly process and assembler.

## 5. Propositional logic, hardware implementation, arithmetic operations

- (a) Propositional logic, well formed formulae, truth values and interpretation of well formed formulae, truth tables.
- (b) Logic and hardware, basic gates (AND, NOT, OR) and their universality, other gates (NAND, NOR, XOR); inverter, half adder, full adder.

## 6. Memory

- (a) Memory - construction of a memory bit using a flip-flop, D-flip-flop and its use in constructing registers.
- (b) Memory organization and access; parity; memory hierarchy - cache, primary memory, secondary memory.

## 7. System and other software

Boot process. Operating system as resource manager, command processing, files, directories and file system. Commonly available programs (editors, compilers, interpreters, word processors, spread sheets, etc.).

## SECTION B

The programming element in the syllabus is aimed at algorithmic problem solving and **not** merely rote learning of Java syntax. The Java version used should be 1.5 or later. For programming, the students can use any text editor and the javac and java programs or any development environment: for example, BlueJ, Eclipse, NetBeans etc. BlueJ is strongly recommended for its simplicity, ease of use and because it is very well suited for an 'objects first' approach.

## 8. Introduction to algorithmic problem solving using Java

### 9. Objects

- (a) Objects as data (attributes) + behaviour (methods or functions); object as an instance of a class. Constructors.
- (b) Analysis of some real world programming examples in terms of objects and classes.

## 10. Primitive values, wrapper classes, types and casting

Primitive values and types: 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.

## 11. Variables, expressions

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

## 12. Statements, scope

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

## 13. Functions

Functions/methods (as abstractions for complex user defined operations on objects), functions as mechanisms for side effects; formal arguments and actual arguments in functions; different behaviour of primitive and object arguments. Static functions and variables. The **this** variable. Examples of algorithmic problem solving using functions (various number theoretic problems, finding roots of algebraic equations).

## 14. Arrays, strings

- (a) Structured data types – arrays (single and multi-dimensional), strings. Example algorithms that use structured data types (e.g. searching, finding maximum/minimum, sorting, solving systems of linear equations, substring, concatenation, length, access to char in string, etc.).
- (b) Basic concept of a virtual machine; Java virtual machine; compilation and execution of Java programs (the javac and java programs).
- (c) Compile time and run time errors; basic concept of an exception, the Exception class, catch and throw.

## SECTION C

### 15. Elementary data structures and associated algorithms, basic input/output

- (a) Class as a contract; separating implementation from interface; encapsulation; private and public.
- (b) Interfaces in Java; implementing interfaces through a class; interfaces for user defined implementation of behaviour.
- (c) Basic data structures (stack, queue, dequeue); implementation directly through classes; definition through an interface and multiple implementations by implementing the interface. Basic algorithms using the above data structures.
- (d) Basic input/output using Scanner and Printer classes from JDK; files and their representation using the File class, file input/output; input/output exceptions. Tokens in an input stream, concept of whitespace, extracting tokens from an input stream (StringTokenizer class).
- (e) Concept of recursion, simple recursive functions (e.g. factorial, GCD, binary search, conversion of representations of numbers between different bases).
- (f) Concrete computational complexity; concept of input size; estimating complexity in terms of functions; importance of dominant term; best, average and worst case.

### 16. Implementation of algorithms to solve problems

The students are required to do lab assignments in the computer lab concurrently with the lectures. Programming assignments should be done such that each major topic is covered in at least one assignment. Assignment problems should be designed so that they are non-trivial and make the student do algorithm design, address correctness issues, implement and execute the algorithm in Java and debug where necessary.

### 17. Social context of computing and ethical issues

- (a) Intellectual property and corresponding laws and rights, software as intellectual property.

- (b) Software copyright and patents and the difference between the two; trademarks; software licensing and piracy.
- (c) Free software foundation and its position on software, open source software, various types of licensing (e.g. GPL, BSD).
- (d) Privacy, email etiquette, spam, security issues, phishing.

## PAPER II - PRACTICAL

This paper of three hours duration will be evaluated internally by the school.

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. After completing the Planning session the candidates may begin with the Examination session. A maximum of 90 minutes is permitted for the Planning session. However, if the candidates finish earlier, they are to be permitted to begin with the Examination session.

### 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 through 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 for the year.

Marks (out of a total of 100) 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) - 20 marks

### Terminal Evaluation

Solution to programming problem on the computer  
- 60 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.)

Viva-voce - 20 marks

(Viva-voce includes questions on the following aspects of the problem attempted by the student: the algorithm and implementation strategy, documentation, correctness, alternative algorithms or implementations. Questions should be confined largely to the problem the student has attempted).

## CLASS XII

*There will be two papers in the subject:*

**Paper I:** Theory- 3 hours ...100 marks

**Paper II:** Practical- 3 hours ...100 marks

### PAPER I-THEORY

*Paper 1 shall be of 3 hours duration and be divided into two parts.*

**Part I (30 marks):** *This part will consist of compulsory short answer questions, testing knowledge, application and skills relating to the entire syllabus.*

**Part II (70 marks):** *This part will be divided into three Sections, A, B and C. Candidates are required to answer **three** questions out of **four** from Section A and **two** questions out of **three** in each of the Sections B and C. Each question in this part shall carry 10 marks.*

#### SECTION A

##### 1. Boolean Algebra

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

- 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).

##### 2. Computer Hardware

- Elementary logic gates (NOT, AND, OR, NAND, NOR, XOR, XNOR) and their use in circuits.
- Applications of Boolean algebra and logic gates to half adders, full adders, encoders, decoders, multiplexers, NAND, NOR as universal gates.

#### SECTION B

The programming element in the syllabus (Sections B and C) is aimed at algorithmic problem solving and **not** merely rote learning of Java syntax. The Java version used should be 1.5 or later. For programming, the students can use any text editor and the javac and java programs or any development environment: for example, BlueJ, Eclipse, NetBeans etc. BlueJ is strongly recommended for its simplicity, ease of use and because it is very well suited for an 'objects first' approach.

### 3. Programming in Java (Review of Class XI Sections B and C)

#### 4. Objects

- (a) Objects as data (attributes) + behaviour (methods or functions); object as an instance of a class. Constructors.
- (b) Analysis of some real world programming examples in terms of objects and classes.

#### 5. Primitive values, wrapper classes, types and casting

Primitive values and types: 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.

#### 6. Variables, expressions

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

#### 7. Statements, scope

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

#### 8. Functions

Functions/methods (as abstractions for complex user defined operations on objects), functions as mechanisms for side effects; formal arguments and actual arguments in functions; different behaviour of primitive and object arguments. Static functions and variables. The **this** variable. Examples of algorithmic problem solving using

functions (various number theoretic problems, finding roots of algebraic equations).

#### 9. Arrays, strings

- (a) Structured data types – arrays (single and multi-dimensional), strings. Example algorithms that use structured data types (e.g. searching, finding maximum/minimum, sorting, solving systems of linear equations, substring, concatenation, length, access to char in string, etc.).
- (b) Basic concept of a virtual machine; Java virtual machine; compilation and execution of Java programs (the javac and java programs).
- (c) Compile time and run time errors; basic concept of an exception, the Exception class, catch and throw.
- (d) Class as a contract; separating implementation from interface; encapsulation; private and public.
- (e) Interfaces in Java; implementing interfaces through a class; interfaces for user defined implementation of behaviour.
- (f) Basic input/output using Scanner and Printer classes from JDK; files and their representation using the File class, file input/output; input/output exceptions. Tokens in an input stream, concept of whitespace, extracting tokens from an input stream (StringTokenizer class).
- (g) Concept of recursion, simple recursive functions (e.g. factorial, GCD, binary search, conversion of representations of numbers between different bases).

### SECTION C

#### Inheritance, polymorphism, data structures, computational complexity

##### 10. Inheritance and polymorphism

Inheritance; base and derived classes; member access in derived classes; redefinition of variables and functions in subclasses; abstract classes; class

Object; protected visibility. Subclass polymorphism and dynamic binding.

### 11. Data structures

- (a) Basic data structures (stack, queue, dequeue); implementation directly through classes; definition through an interface and multiple implementations by implementing the interface. Basic algorithms using the above data structures.
- (b) Recursive data structures: singly and doubly linked lists, binary trees, tree traversals, binary search tree. Algorithms using these data structures (merge sort and quick sort, searching).

### 12. Complexity and big O notation

Concrete computational complexity; concept of input size; estimating complexity in terms of functions; importance of dominant term; 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.).

### 13. Implementation of algorithms to solve problems

The students are required to do lab assignments in the computer lab concurrently with the lectures. Programming assignments should be done such that each major topic is covered in at least one assignment. Assignment problems should be designed so that they are non-trivial and make the student do algorithm design, address correctness issues, implement and execute the algorithm in Java and debug where necessary.

## PAPER II - PRACTICAL

This paper of three hours duration will be evaluated by the Visiting Examiner appointed locally and approved by the Council.

The paper shall consist of three programming problems from which a candidate has to attempt any one. The practical consists of the two parts:

- (3) Planning Session
- (4) Examination Session

The total time to be spent on the Planning session and the Examination session is three hours. After completing the Planning session the candidates may begin with the Examination session. A maximum of 90 minutes is permitted for the Planning session. However, if the candidates finish earlier, they are to be permitted to begin with the Examination session.

#### 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 the assignments done as part of the practical work through 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 for the year.

Marks (out of a total of 100) 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) - 10 marks

## **Terminal Evaluation**

Solution to programming problem on the computer  
- 60 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.)

Viva-voce - 20 marks

(Viva-voce includes questions on the following aspects of the problem attempted by the student: the algorithm and implementation strategy, documentation, correctness, alternative algorithms or implementations. Questions should be confined largely to the problem the student has attempted).

## **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 board with white board markers should be available.
- A fully equipped Computer Laboratory that allows one computer per student.
- Internet connection for accessing the World Wide Web and email facility.
- The computers should have a minimum of 256 MB (512MB preferred) RAM and a PIII or higher processor. The basic requirement is that it should run the operating system and Java programming system (Java compiler, Java runtime environment, Java development environment) at acceptable speeds.
- Good Quality printers.

## **Software:**

- Any suitable Operating System can be used.
- JDK 1.5 or later.
- Documentation for the JDK version being used.
- A suitable text editor. A development environment with a debugger is preferred (e.g. BlueJ, Eclipse, NetBeans). BlueJ is recommended for its ease of use and simplicity.