

# COMPUTER SCIENCE (71)

## Aims:

1. To enable candidates to comprehend the concepts and practices of Computer science.
2. To develop an understanding of how computers store and process data.
3. To enable candidates to describe the major components of computer hardware, their functions and interaction.
4. To develop an understanding of the fundamental concepts of programming and the ability to apply the same.
5. To develop an appreciation of the implications of computer use in contemporary society.

## CLASS IX

There will be **one** paper of **two** hours duration carrying 80 Marks and Internal Assessment of 20 marks.

The paper will be divided into **two** Sections A and B.

**Section A** (20 marks): This section will consist of compulsory short answer questions, testing knowledge, application and skills relating to elementary/fundamental aspects of the entire syllabus.

**Section B** (60 marks): This section will consist of questions based on programming. There will be a choice of questions and candidates will be required to answer **four** questions from this section.

### PART I -THEORY

#### 1. Computer hardware: parts of a computer and their functions

CPU, the clock, cache memory, primary memory, secondary memory, input and output devices, communication devices (the aim is not to describe/discuss an exhaustive list of devices but to understand what parts are present in a typical computer and what the function of each part is).

*Teachers can open a computer and show the various parts; explain how the motherboard becomes a kind of 'central coordinator' where all the others link up; point out the various chips on the motherboard that are responsible for the different functions - CPU, memory, clock, boot ROM, etc.*

*Similarly, it is good to show students a floppy disk and hard disk from the inside (an old non-functional disk can be used for this purpose).*

*Peripheral devices should also be shown from the 'inside', if possible.*

#### 2. Data representation and internal computer structure

- (i) Number systems, base of a number system - decimal, binary, octal, hexadecimal representation, conversion between various representations, character representations (ASCII, ISCII, Unicode).
- (ii) Representations for integers, real numbers, limitations of finite representations.
- (iii) Internal structure of a computer, a simple decimal load and store computer and its machine language, instruction format, registers, program counter, instruction register; register addressing modes, instruction cycle, assembly language for the same computer, simple algorithms in assembly language.

*The teachers must review the place notation for decimal numbers, then make students count and do arithmetic (addition and subtraction) with base 2 and 8. This develops intuition for conversion of numbers from one base to another. Emphasize the finiteness of representations when only limited space is available - a bit, byte and word can be introduced at this stage to talk about sizes. Give examples to enable students to understand the maximum and minimum sized numbers that can be represented in a given number of bits. Students can write simple programs to keep increasing the value of an integer till it overflows and determine the number of bits to store numbers of that type. Discuss different ways to represent negative*

numbers (signed magnitude, ones complement and twos complement). Introduce sign, mantissa, radix, exponent notation and how real numbers can be represented ( $\text{sign} * \text{mantissa} * \text{radix}^{\text{exponent}}$ ). Discuss normalized and non-normalized representations, 32-bit and 64-bit representations. In (i) it is useful to introduce coding systems for other languages - like ISCII (for Indian languages) and Unicode as a standard for all languages of the world. In (iii) a simple decimal computer simulator can be used which has load, store, arithmetic, simple conditional jumps, jump instructions, simple input output. The idea is to give a clear understanding of how a typical computer works, without going into too much detail. The student can write simple programs using the instruction set of the machine so that they understand the need for high level languages. This will also clarify the basic idea of a stored program where program is treated as data.

### 3. Computer software

The boot process, operating system (resource management and command processor), file system.

- (i) Boot process, operating systems - resource management, command processing.
- (ii) Directories, files and hierarchical file system.
- (iii) Programming languages (machine language, assembly language, high level language).
- (iv) Compilers and interpreters.
- (v) Application software.

One natural way to visualize an OS is as a software layer which creates a virtual machine that is much more useable than the bare machine. This involves giving the user a high level command interface and the management of the raw machine resources (like memory, CPU etc.) so that they can be used efficiently. The languages at different levels (machine, assembly, higher) can be motivated by a discussion based on the contents of 2 (iii) above and topic 6 below. Application software is best introduced through application software that the student will be using like browsers, spreadsheets, word processing etc., this can be integrated with the discussion in topic 7.

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

- (i) Intellectual property and corresponding laws and rights, software as intellectual property.
- (ii) Software patents, copyrights, and trademarks, software licensing and piracy.
- (iii) Free software foundation and its position on software, open source software.
- (iv) Privacy, email etiquette.

*There can be very interesting discussions in the class regarding the ethical issues. There can be discussions on copyright, fair use, a program as free speech and Digital Millennium Copyright Act. The students can gather more information from the net. The stress should be on following good etiquette and ethical practices.*

### 5. Algorithms

- (i) Concept of an algorithm.
- (ii) Properties of an algorithm (finite, definite, terminating, precise).
- (iii) Basic ideas of the complexity of an algorithm - space complexity, time complexity.

*A number of problems should be introduced to familiarize the student with the idea of various ways in which operations on data yield solutions to problems. (Please refer to topic 6).*

*The problems should use different forms of data - numeric, nonnumeric, structured.*

*Students should be asked to focus upon what are the outputs required, the inputs needed and work out the solutions to the problems.*

*Informal structured English can be used to write the solutions.*

*Students should be asked to visualize sample data for the problem especially for the extreme cases.*

*They should be asked to trace the algorithms to see if the expected output is obtained.*

*This would help stabilize the concept of algorithms.*

*Simple algorithms for number problems can be discussed here. These can be coded in the programming language that is covered as part of topic 6. Simple concrete complexity can be discussed so that students understand that not all algorithms are the same with respect to time and*

*space complexity. Also, briefly discuss space-time tradeoffs.*

## 6. Programming Using a High Level Language

The programming element in the syllabus is aimed at problem solving and **not** on merely rote learning of the commands and syntax of particular programming languages. Students have the option to use either BASIC or C++ in order to implement the high level language concepts and algorithms and to use them for solving problems. While choosing BASIC care must be taken to choose a standard version that has “block if structures”, “functions through which parameters may be passed and values returned”. **Very old versions using “goto statements” must not be used.** Care must be taken that ‘**standard and recent**’ versions of the languages are used on the computer. It is recommended that students mention the version of the language being used while writing answers in order to avoid ambiguity. For example, software such as Microsoft Quick BASIC, Borland Turbo C++, Visual C++ or GNU C++ on Linux can be used.

*The emphasis here should be on problem solving. The design approach here may vary. The users of QBASIC should use the structured programming approach while C++ users may use the object-oriented approach.*

*It must be remembered that the language (QBASIC/ C++) is just a vehicle for expressing solutions.*

*The object-oriented techniques are recommended as students learn these very naturally and quickly. Once learnt they are very easy to use.*

*Simple demonstration programs can be executed on the computer to illustrate various concepts as they are introduced.*

- (i) Primitive data types supported by the language (integers, floating point numbers, characters, booleans etc. - will depend on the language), variables (and their declaration - based on language), assignment, difference between the left-hand side and right-hand side of an assignment.
- (ii) Expressions - arithmetic and logical, evaluation of expressions, type of an expression (depends on language). Operators, associativity and precedence of operators.

- (iii) Statements, blocks (where relevant), scope and visibility of variables.
- (iv) Conditional statements (if and if-then-else), switch, break, default.
- (v) Loops (for, while-do, do-while).
- (vi) Simple input/output using standard input/output.

*The teachers should introduce problem solving through numerous examples and informally familiarize the students with the idea of various ways in which operations on data yield solutions to problems. The examples should use different forms of data (numeric, non-numeric, structured). In the beginning the solutions should be written in a freely invented structured form of English. The informal structured English constructs should not be too high level – they should be at a level where they can be unambiguously carried out – which means they are at par with programming language constructs. For example, primitive constructs like minimum or maximum of a set of numbers, sort etc. should not be allowed (see examples below). Such compound constructs should be introduced as abstractions, that is as functions or procedures. In the process of writing the solutions, motivate and informally introduce:*

- *How the real world presents us with different types of data (numeric, non-numeric, boolean, structured).*
- *The notion of using a variable to hold data.*
- *How the assignment operation is used to change the data a variable denotes.*
- *How operations on the variable actually operate on the data.*
- *How input and output are needed.*
- *How the sequence of operations on data can be abstracted out (as an algorithm) and be repeated on different data sets.*
- *The concept of a processor (the teacher) and a store (the blackboard) by mechanically tracing/executing the solutions.*
- *How the same kind of repetitive operation sequences seem to appear again and again in the solutions (conditionals, loops).*
- *How some solutions can be reused in solving other problems.*

*Throughout this topic the informal structured English constructs of algorithms should be shown to correspond to similar constructs in the language. Programs should be written for all the examples. Students should run all the programs discussed in class in the lab. Some of these programs will be done only after the necessary concepts have been introduced.*

*Sample examples:*

- a) Multiplication as repeated addition.*
- b) Finding if a number is a prime number.*
- c) Find the maximum or minimum of 3 numbers, 10 numbers, a given set of n numbers (requires input/output).*
- d) Ordering (ascending or descending) a set of 3 numbers; a set of 10 numbers; a given set of n numbers. Try to reuse what is done in c).*
- e) Finding the number of vowels in a given sentence (composite data, non-numeric data).*
- f) Finding the number of words in a given sentence.*
- g) Finding out who has got the maximum aggregate marks in the class after an exam in all the subjects (structured data, accessing elements within structured data).*

## **7. Computers in everyday life**

- (i) Familiarity with software for word processing, databases, spreadsheets, making presentations.

- (ii) Basic introduction to the Internet, browsing, email.

*Students should be encouraged to use computers to write the assignments, project reports, create banners and placards for school events. They will automatically learn to use the word processors and spreadsheets, etc.*

*Students should be encouraged to log on to the Internet to gather material for their projects. A number of interesting assignments can also be given in this section.*

## **PART II - INTERNAL ASSESSMENT (PRACTICAL WORK)**

Part II (Practical work) will carry 20 marks and shall be assessed on a continuous basis throughout the year. The assessment of practical work should include small projects using software in item 7 and solutions to programming problems in item 6, which have been coded and run in the higher level language being used in the course.

Teachers should maintain a record of work done through the year and give it due credit at the time of cumulative evaluation at the end of the year.