CHAPTER 7 Software Concepts

Software is a general term which is used to describe the instructions that are given to computer These instructions can be either a single programme or a group of programmes.

TYPES OF SOFTWARE

Software is generally classified into three specific categories in the computer world:

1. System software
2. Application software
3. Utility software

1. System software : System software is computer software designed to operate the computer hardware and to provide a platform for running application software. System software is composed of operating system and system utilities. These two major components have great importance on acquiring any computer system. The more you know about operating system the better your computer will serve you. This is why operating system and system utilities have a key role on acquiring a computer system.

- The computer BIOS and device firmware, which provide basic functionality to operate and control the hardware connected to or built into the computer.
- The operating system (prominent examples being Microsoft Windows, Mac OS X and Linux), which allows the parts of a computer to work together by performing tasks like transferring data between memory and disks or rendering output onto a display device. It also provides a platform to run high-level system software and application software.
- Utility software, which helps to analyse, configure, optimize and maintain the computer.

2. Application software : These programmes are developed by the user in order to perform some specific function for the organisation. For example, a payroll system to compute the salaries of the employees of an organisation is termed as an application software.

3. Utility software : Utility software is a kind of system software designed to help analyze, configure, optimize and maintain the computer. A single piece of utility software is usually called a utility or tool.

Utility software should be contrasted with application software, which allows users to do things like creating text documents, playing games, listening to music or surfing the web. Rather than providing these kinds of user-oriented or output-oriented functionality, utility software usually focuses on how the computer infrastructure (including the computer hardware, operating system, application software and data storage) operates. Due to this focus, utilities are often rather technical and targeted at people with an advanced level of computer knowledge.

Most utilities are highly specialized and designed to perform only a single task or a small range of tasks. However, there are also some utility suites that combine several features in one piece of software. Most major operating systems come with several pre-installed utilities

Example: disk cleaner, disk fragmenter, disk compression, file manager.

COMPUTER PROGRAMMING

Computer programming (often shortened to programming or coding) is the process of designing, writing, testing, debugging / troubleshooting, and maintaining the source code of computer programs. This source code is written in a programming language. The purpose of programming is to create a program that exhibits a certain desired behaviour. The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

PROGRAMMING LANGUAGES

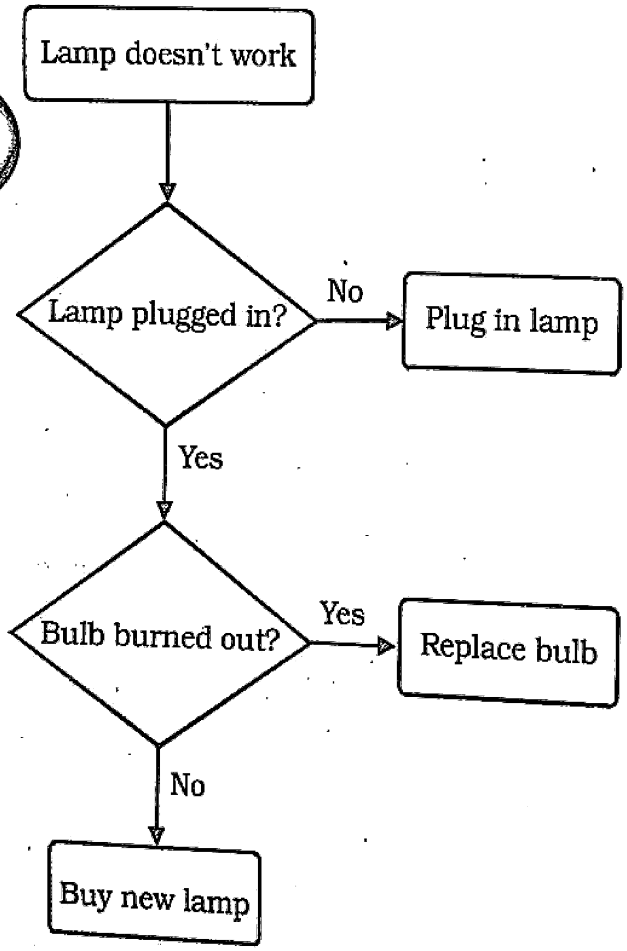
A programming language consists of words, symbols and usage rules pertaining to the grammar that permits people to communicate with the computer. Understanding of computer software is imperfect without a basic knowledge of programming languages. Programming languages allow the programmers and end-users to develop the programmes that are executed by the computer. Many programming languages exist in the world today. Each one of the languages have their own unique vocabulary, grammar and usage. Some of these languages have been created to serve a special purpose while others are more flexible and general purpose and are suitable for many types of applications. However in general, programming languages must cater to the following tasks :

— input/output
— text manipulations/calculations
— logic/comparison
— storage/retrieval

FLOWCHART

A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Data flows are not typically represented in a flowchart, in contrast with data flow diagrams; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Flowcharts typically use standard symbols to represent different stages or actions within the chart. For example, each step is shown within a rectangle, while each decision is displayed in a diamond. Arrows are placed between the different symbols to show the direction the process is flowing. While flowcharts can be created with a pen and paper, there are several software programs available that make designing flowcharts especially easy. Common programs that can be used to create flowcharts include Smart Draw and Visio for Windows and Omni raffle for the Mac.



CLASSIFICATION OF PROGRAMMING LANGUAGE

Machine Languages : Machine language is the lowest form of computer language. Programmes were only written in binary based machine level language in the first generation computers. The computer understands this language only at its lowest level.

An instruction prepared in machine language has two parts :

1. Op-code : This is the first part and is the command or operation and it tells the computer what function to perform.

2. Operand : The second part of the instruction is the operand and it tells the computer where to find or store the data or instructions that are to be manipulated. The number of operands in an instruction varies from computer to computer. In a single operand machine, the binary equivalent of "ADD 0481" could cause the value in a storage location 0481 to be added to a value stored in the arithmetic & logic unit. The single operand format is popular in the smallest microcomputers whereas the two operand structure is found in most other machines.

The set of instructions in a machine level language can be divided into four categories :

1. Arithmetic- add, subtract, multiply and divide
2. Controlled- load, store, jump instructions
3. Input output – Read and write
4. Direct use – Halt, start and end

No arithmetic or comparison operations are done in the primary memory of the computer. Instead it is done in the ALU's special register called accumulator. Thus if we need to add two numbers, we require one instruction which will order the control unit to place a number in the accumulator and another instruction to identify the operation of addition.

Symbolic/Assembly Languages : In order to reduce the burden, symbolic languages, commonly known as assembly languages was developed in 1950's for the second generation computers.

This language permits the use of symbols or mnemonics which are two or three letter abbreviations for the function to be performed by the instruction. These are then translated by using symbolic equivalence table. Assembly language has many of the same features to control registers etc. However, the disadvantage of using binary has been removed.

Functions of Assembler

(i) The Assembler translates the function code into its machine code equivalent.

(ii) It assigns absolute addresses to any symbolic address or label names.

(iii) It places each instruction in central memory.

(iv) It identifies indirect addresses from direct addresses and sets the appropriate bit in the address portion of the instruction.

(v) It checks the syntax of each instruction and generates error messages.

(vi) It provides, optionally, a cross reference table between all symbolic names and their absolute addresses.

(vii) It informs the control unit to execute the program after all errors have been corrected.

Advantages of Assembly languages

(i) They save time and reduce detail as compared to machine language.

(ii) Lesser number of errors are made and errors are easier to detect.

(iii) Assembly programs are easier to modify than machine language programs.

Disadvantages of Assembly Language

(i)Writing a code is time consuming.

(ii) Assembly languages are machine dependent.

HIGH LEVEL LANGUAGES

The disadvantages of using assembly language brought about the development of higher level languages. Unlike the assembly programs, high level language programs may be used with different makes of computers with little modification. High level languages are easier to learn than symbolic languages; they require less time to write, are easier to maintain, provide better documentation and 4 or 5 low-level instructions are reduced to a single high level statement. Some of the popular high level languages are given in the table below.

Language	Meaning	Main Application Area
FORTRAN	Formula Translator	Scientific & Engineering
COBOL	Common Business	Oriented language Commercial
ALGOL	Algorithmic Language	Scientific
RPG	Report Generator	Commercial
APL	A Programming Language	The Sharing System
PL/1	Programming Language 1	Scientific/Commercial
BASIC	Beginners All purpose Code	Symbolic Instruction Teaching
PASCAL	Named after the French Philosopher	Teaching

CLASSIFICATION OF HIGH LEVEL LANGUAGES

High level languages are sometimes classified as :

1. Procedure oriented languages,
2. Problem oriented languages and
3. Interactive programming languages.

Procedure Oriented Languages : Procedure oriented languages provides easy to use features which allow the programmer to write processing steps required for a particular application. Examples of procedure oriented languages are COBOL, FORTRAN, PL/1 etc.

Problem Oriented Languages : These languages attempt to solve processing requirements with minimal programming effort; thereby allowing the user to concentrate on the desired results, rather than individual steps needed to get these results. A typical example of a problem oriented language is RPG.

Interactive Programming Languages : These languages have features which allow the user to interact with the program in a conversational fashion. Typical example in BASIC language.

Fourth Generation Languages : Computer people refer to the machine language era as the first generation, the symbolic codes as the second generation and the advent of the high level languages as the third generation. Successive generation languages brought about an increase in the programmers' productivity. These generations of software have now been followed by emergence of software vendors offering a variety of application development tools aimed at further productivity improvements. These tools are collectively known as 4th generation languages (4GL's).

Essentially, a 4GL tool interacts with DBMS software to store, manipulate and retrieve the data required to satisfy user requirement. In contrast to a high level language, which is procedural, a 4GL is non-procedural language in the sense that it allows the user to simply specify what the output be without describing all the details of how data is to be manipulated to get the results.

A clear cut definition of a 4GL is not possible because each tool or language is vendor specific. In general, however, we have the following classification :

(i) End user oriented 4GLs : These are designed for applications that process low data volumes and run on mainframes and may be used by users or programmers. They have their own internal data base management software which in turn interacts with the organisation's DBMS. Non-computer professionals use these products to query databases and develop their own applications and generate reports with minimum training.

(ii) These tools are offered by the same suppliers who also supply the complex DBMS packages to the organisation. These tools offer a larger set of commands than the end user 4GLs. They are also able to handle larger volume of data and are generally used by programming specialists. These include query languages, application generators and report generators.

The following chart depicts some of the major differences between higher level languages and 4GLs.

	Third Generation Languages	4GLs
(i)	Used by professional Programmers	May be used by non-programmers also
(ii)	Require task performance specification (how)	Require specification of what task to perform (what)
(iii)	All alternatives are specified	Default alternatives are built in
(iv)	Requires large no. of instructions	Require far fewer instructions
(v)	Code difficult to read, understand	Code easy to read and maintain
(vi)	Originally developed for batch	Developed primarily for online
(vii)	Can be difficult to learn	Easy to learn
(viii)	Difficult to debug	Errors easier to locate because of shorter programs and use defaults
(ix)	Typically file oriented	Typically database oriented

SOURCE CODE

Source code is text written in a computer programming language. Such a language is specially designed to facilitate the work of computer programmers. Initially, a programmer writes a program in a particular programming language. This form of the program is called the source program, or more generically, source code. To execute the program, however, the programmer must translate it into machine language, the language that the computer understands. The first step of this translation process is usually performed by a utility called a compiler. The compiler translates the source code into a form called object code. Sometimes the object code is the same as machine code; sometimes it needs to be translated into machine language by a utility called an assembler. Source code is the only format that is readable by humans. When you purchase programs, you usually receive them in their machine-language format. This means that you can execute them directly, but you cannot read or modify them. Some software manufacturers provide source code, but this is useful only if you are an experienced programmer.

OBJECT CODE

The code produced by a compiler. Programmers write programs in a form called source code. The source code consists of instructions in a particular language, like C or FORTRAN. Computers, however, can only execute instructions written in a low-level language called machine language. To get from source code to machine language, the programs must be transformed by a compiler. The compiler produces an intermediary form called object code. Object code is often the same as or similar to a computer's machine language. The final step in producing an executable program is to transform the object code into machine language, if it is not already in this form. This can be done by a number of different types of programs, called assemblers, binders, linkers, and loaders.

Object Oriented Languages : Object Oriented Programming (OOPS) have been around since the 1960's. However, it was only in the eighties that object oriented languages became a major consideration in software development. This was due to the increasing emphasis of users to provide an easier way of doing their computing in which the user wants all his needs to be fulfilled through windows, buttons and menus (all objects).

The basic philosophy of object oriented languages is to encapsulate the data and procedures (methods) together into an object so that the conventional method of separating the data from the procedures is eliminated and the programmer need not keep track of what is happening to the data each time a program was run. Once objects are programmed, they are reusable. Languages like C, C++, JAVA are examples of object oriented languages and have become today's programming languages by choice.

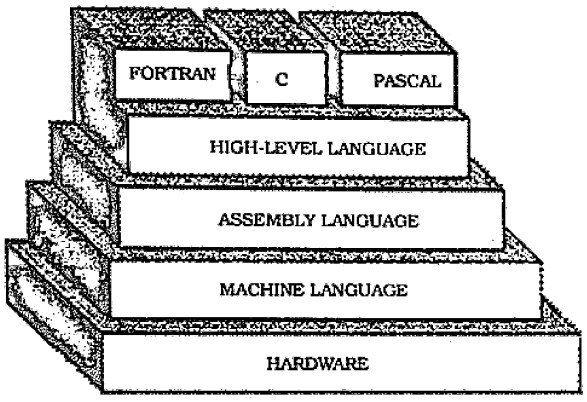
Language Translator Programs : These are programs that translate programs written in other languages into machine language instructions code, which the computer can execute. There is also a programmer to write his program by providing creation and editing facilities.

Translation programs are known by various names :

(i) Assemblers : An assembler translates the symbolic instruction code of programs written in assembly language into machine code.

(ii) Interpreter : An interpreter translates and executes each program statement one at a time, instead of producing a complete machine language program like assemblers and compilers do.

Compiler simply converts the source code into machine code, which can be run directly by the operating system as an executable program. Interpreters bypass the compilation process and execute the code directly. Since interpreters read and execute code in a single step, they are useful for running scripts and other small programs



With an interpreter, the source program is not assembled into an object program. The results are computed immediately after an instruction has been translated. This process allows very efficient use of computer and programmer time during the debugging of the application.

The interpreter need not be a software which is loaded from external sources. It may be permanently residing in a ROM program. Microsoft BASIC interpreter uses such a program. Some of the Interpretive languages are BASIC, LISP, FORTH, APL etc.

(iii) Compilers : A compiler is a program which produces a machine level program from the specifications of a high level language by generating one or more than one machine instruction for each high level instruction, i.e. it translates the higher level program into machine code.

The compilation process has several phases :

1. Parsing : Parsing is the process of breaking down

(i) a statement grammatically in terms of its functions, syntactic relations etc.

(ii) the statements of source language are examined and analysed.

(iii) the data dictionary comprising of data descriptions is built.

(iv) a coded representation of the statements is created.

(v) a copy of the source program is printed.

(vi) if necessary, appropriate diagnostic messages are issued.

2. Mapping : In the second phase, a mapping process is performed. The dictionary entries are examined, and the declarations are Mapped into the data formats of the object computer.

3. Generation of object code : In the final phase, appropriate object code is generated to handle the expressed operation.

ALGORITHM

An algorithm is a set of instructions, sometimes called a procedure or a function that is used to perform a certain task. This can be a simple process, such as adding two numbers together, or a complex function, such as adding effects to an image. For example, in order to sharpen a digital photo, the algorithm would need to process each pixel in the image and determine which ones to change and how much to change them in order to make the image look sharper.

Most computer programmers spend a large percentage of their time creating algorithms. (The rest of their time is spent debugging the algorithms that don't work properly.) The goal is to create efficient algorithms that do not waste more computer resources (such as RAM and CPU time) than necessary. This can be difficult, because an algorithm that performs well on one set of data may perform poorly on other data.

GENERATORS

The generator is a routine that performs a creative function, for example a report generator, a program generator. A report generator creates a report requiring little or no calculations or complex logic. In this process the programmer writes specifications for the reports rather than the program instructions necessary to generate the report. The specifications may be the print positions in which the data fields have to be printed, totals required etc. The specifications are subsequently converted into a set of specifications and interfaced with the report generator processor. The report generator then produces a report as per the format specified.

THE EDITOR

Editor is the most often used systems program. These are interactive programs that are stored in the memory and allow the user to write to program, generate text or to make the modifications to either of these. The editor is a systems program that is typically stored on the hard disk and whenever needed is called into the RAM. All source programs for assemblers and compilers are typically written with the editor.

THE LOADER

The loader is a short simple program that is used to load a program into the computer's memory. The loader program is a sequence of instructions that

transfer the program from the hard disk/floppy to the memory. The loader is typically permanently stored in the ROM of the computer.

PROGRAM LIBRARY

A library is a collection of programs which can be searched by a linking loader for requested programs. The loader first loads all the user's supplied programs and then searches the library to find program names that match unsatisfied external references. Systems may allow any number of libraries to be specified. A linking loader will search each one in turn.

THE MONITOR

Almost all of microcomputers use some form of monitor. Typically it is stored in ROM and performs some of performing functions :

(i) enter or load instructions and data.

(ii) change contents of memory locations and registers.

(iii) display contents of memory locations and registers.

(iv) execute programs.

FIRMWARE

Firmware is a software program or set of instructions programmed on a hardware device. It provides the necessary instructions for how the device communicates with the other computer hardware. But how can software are programmed onto hardware? Good question. Firmware is typically stored in the flash ROM of a hardware device. While ROM is "read-only memory," flash ROM can be erased and rewritten because it is actually a type of flash memory. as most embedded devices contain firmware at more than one level. Subsystems such as CPUs, flash chips, communication controllers, LCD modules, and so on, have their own (usually fixed) program code and/or microcode, regarded as "part of the hardware" by the higher-level(s) firmware.

DEBUGGING

Errors caused by faulty logic and coding mistakes are referred to as "bugs." Finding and correcting these mistakes and errors that prevent the program from running and producing correct output is called "debugging.

SOFTWARE TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

1. meets the business and technical requirements that guided its design and development;
2. works as expected; and
3. Can be implemented with the same characteristics.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

SOFTWARE QUALITIES

When we talk about the qualities expected from the software the list becomes very long. These qualities can be expected by the user, by the producer or by the leader of the project. The qualities can also be expected from both the product and the production process.

The user would like the product to be reliable, easy to use, efficient and accurate whereas the producer wants it to be maintainable, portable, extensible and verifiable. At the same time the manager of the software project wants the software project process to be productive and easily controllable.

The qualities of software can broadly be dividend into :

(i) External vs Internal Qualities

(ii) Product vs Process Qualities

(i) External : The qualities a software should possess can be divided into internal and external qualities. Internal qualities deal with the qualities from the developers point of view whereas the external qualities are the qualities expected by the user. The internal qualities are the ones which are responsible for the structure of the software and infact these are the ones which help developers in achieving the external qualities. Users are generally not bothered about the internal qualities whereas the developers concentrate on internal qualities to achieve the target of external qualities e.g., the internal quality of verifiability is responsible for achieving the external quality of reliability.

(ii) Product : The product and process qualities are also very closely related as in some cases of internal and external qualities. There are some properties that apply to both product and process. For example, the careful planning of the test data in the process of design and development of the product, will increase the reliability of the product, whereas the quality like efficiency is expected from both product and process.

REPRESENTATIVE QUALITIES

We would now talk about the qualities of software products as well as the process. We call them "Representative Qualities". We would be analysing, the "RQ" in terms of the classifications discussed above :

1. Correctness, Reliability, and Robustness : A program is said to be correct if it functions as per the specifications or the requirement specification. That means it is doing whatever was expected out of it.

A software is said to be reliable if the user can depend upon it. A reliable software is defined in terms of probability i.e., the software will function (as expected) for a specified time interval.

A programme is said to be "robust" if it behaves "reasonably" even in the unanticipated circumstances. For example, when a programme encounters incorrect input or some other problem and it generates an uncoverable run time error as soon as the function is performed, then the programme cannot be called robust.

2. Performance : Any engineering product lays stress on performance of the product. But in software engineering, the performance is expected with efficiency. A software system can be called efficient only if it uses the resources economically.

3. User Friendliness : A system is said to be user friendly if the users find it "easy to use". This "ease of use" may vary from user to user. A novice might find the verbal messages user friendly, an expert programmer ignores them without even reading them, similarly a new programmer or a non programmer may find the use of commands very difficult whereas a programmer may find the use of commands easier.

4. Verifiability : This property holds true if its properties can be verified easily. As correctness and performance are the properties which we would like to verify, the properties can be verified by means of analysis or testing.

5. Maintainability : By "software maintenance" we generally mean the modifications required after the release of the software in the market.

Earlier maintenance meant only removing the errors and the bugs. A considerable amount is spent on this simple word "bug". But now maintenance adds on this "bugs fixing". It includes the additions or enhancements to the products which were introduce there when the project started or which were incorrectly stated.

The word "maintenance" does not go with word "software". By maintenance we generally mean the maintenance of wear and tear of the product. But in software there is nothing termed as wear and tear. But since this is the "in" thing how, we area forced to use it.

6. Reusability : Reusability is quite similar to evolvability. In evolvability modifications are made in the product to take out a new version of the product whereas in reusability the same product is reused with some modifications. Although it seems easy but it is very difficult to build variable products. While reusability is an important for reducing software production costs examples of reusable software are very rare.

Reusability is applied to the process as well. The same process can be reused for building up different products.

7. Portability : Software is said to be portable if it can be run in different environment. By "environment" we mean the hardware platform or a software environment such as a particular operating system.

Portability refers to the ability to run a system on different hardware platforms. As the money spent on software and hardware increases, portability gets more importance.

8. Understandability : Some software systems are easy to understand whereas others are not.

This is an internal product quality and helps in achieving evolvabilty and verifiability. While taking it as an external quality, it is a component of user friendliness.

9. Productivity : This is a quality of software production process and is used to measure the efficiency of the process. An efficient process results in faster delivery of the product.

10. Timeliness : It is again a process-related quality which talks of-timely delivery of software. This timeliness is generally lacking in software production process.

11. Visibility : A software development process in said to be visible if the documentation of all its steps is done properly and clearly. "Transparency" and "openness" can be inter-changed with the term visibility. It is both an internal and external quality. Visibility not only requires the documentation of all the processes but the current status of intermediate products also. A product is visible if it is clearly structured such as a collection of modules, with clearly understandable functions and easily accessible documentation.

PROGRAMMING LANGUAGES

A large number of high level languages have been developed for specific requirements. Some of the commonly used languages for various applications are :

I. Scientific and Engineering :
BASIC FORTRAN

II. Business and Commercial purpose
COBOL BASIC

III. Text Processing
Lisp SOSL

IV. General Purpose
PASCAL ADA
C

V. Artifical Intelligence (AI)
PROLOG LISP

OBJECTIVE QUESTIONS

1. Word processing, spreadsheet, and photo-editing are examples of :
(1) application software.
(2) system software.
(3) operating system software.
(4) platform software.
(5) None of these

2. _____ is a set of computer programs used on a computer to help perform tasks.
(1) An instruction
(2) Software (3) Memory
(4) A processor
(5) None of these

3. System software is the set of programs that enables your computer's hardware devices and _____ software to work together.
(1) management
(2) processing
(3) utility (4) application
(5) None of these

4. The two broad categories of software are:
(1) word processing and spreadsheet.
(2) transaction and application.
(3) Windows and Mac OS.
(4) system and application
(5) None of these

5. Personnel who design, program, operate and maintain computer equipment refers to
(1) Console-operator
(2) Programmer
(3) Peopleware
(4) System Analyst
(5) None of these

6. A computer program that converts an entire program into machine language is called a/an
(1) Interpreter (2) Simulator
(3) Compiler (4) Commander
(5) None of these

7. A computer program that translates one program instructions at a time into machine language is called a/an
(1) Interpreter (2) CPU
(3) Compiler (4) Simulator
(5) None of these

8. programs designed to perform specific tasks is known as
(1) system software
(2) application software
(3) utility programs
(4) operating system
(5) None of these

9. Each model of a computer has a unique
(1) Assembly of a computer
(2) Machine language
(3) High level language
(4) All of the above
(5) None of these

10. On a PC, how much memory is available to application software?
(1) 1024 KB (2) 760 KB
(3) 640 KB (4) 560 KB
(5) None of these

11. The language that the computer can understand and execute is called
(1) Machine language
(2) Application software
(3) System program
(4) All of the above
(5) None of these

12. The steps and tasks needed to process data, such as responses to questions or clicking an icon, are called:
(1) instructions
(2) the operating system.
(3) application software.
(4) the system unit.
(5) None of these

13. Which statement is valid about computer program?
(1) It is understood by a computer
(2) It is understood by programmer
(3) It is understood user
(4) All of the above
(5) None of these

14. Software in computer
(1) Enhances the capabilities of the hardware machine
(2) Increase the speed of central processing unit
(3) Both of above
(4) None of above
(5) None of these

15. Which of the following is not computer language?
(1) High level language
(2) Medium level language
(3) Low level language
(4) All of the above
(5) None of these

16. Which language is directly understood by the computer without translation program?
(1) Machine language
(2) Assembly language
(3) High level language
(4) None of above
(5) None of these

17. Instruction in computer languages consists of
(1) OPCODE
(2) OPERAND
(3) Both of above
(4) All of the above
(5) None of these

18. Machine language is
(1) Machine dependent
(2) Difficult to program
(3) Error prone
(4) All of the above
(5) None of these

19. The translator program used in assembly language is called
(1) Compiler (2) Interpreter
(3) Assembler
(4) Translator
(5) None of these

20. Easily understood language is
(1) Machine language
(2) Assembly language
(3) High level language
(4) Medium level language
(5) None of these

21. Which of the following is called low level languages?
(1) Machine language
(2) Assembly language
(3) Both of the above
(4) Either 1 or 2
(5) None of these

22. Which of the following is problem oriented language?
(1) High level language
(2) Machine language
(3) Assembly language
(4) Low level language
(5) None of these

23. A compiler is a translating program which

(1) Translates instruction of a high level language into machine language
(2) Translates entire source program into machine language program
(3) It is not involved in program's execution
(4) All of the above
(5) None of these

24. Which of the following is machine independence program?
(1) High level language
(2) Low level language
(3) Assembly language
(4) Machine language
(5) None of these

25. Which statement is valid about interpreter?
(1) It translates one instruction at a time
(2) Object code is saved for future use
(3) Repeated interpretation is not necessary
(4) All of above
(5) None of these

26. Which is the limitation of high level language?
(1) Lower efficiency
(2) Machine dependence
(3) machine level coding
(4) None of above
(5) None of these

27. High level language is also called
(1) Problem oriented language
(2) Business oriented language
(3) Mathematically oriented language
(4) All of the above
(5) None of these

28. FORTRAN is
(1) File Translation
(2) Format Translation
(3) Formula Translator
(4) Floppy Translation
(5) None of these

29. Which of the following is NOT a computer programming language?
(1) C (2) C++
(3) Java (4) COBOL
(5) Microsoft

Bank of Baroda Clerk Exam, 30.11.2008

30. What kind of software would you most likely use to keep track of a billing account ?
(1) Word processing
(2) Electronic publishing
(3) Spreadsheet
(4) Web authoring
(5) None of these

Bank of Baroda Clerk Exam, 30.11.2008

31. What is correcting errors in a program called ?
(1) Interpreting
(2) Translating
(3) Debugging (4) Compiling
(5) None of these

SBI PO Tier-I Exam, 13.10.2008

32. _____ software allows users to perform calculation on rows and columns of data.
(1) Word processing
(2) Presentation graphics
(3) Database Management System
(4) Electronic spreadsheet
(5) None of these

Allahabad Bank Clerk Exam, 31.08.2008

33. In UNIX, command "!$" is used to
(1) repeat last word of last command line
(2) repeat entire last command line
(3) count the number of arguments
(4) match to unknown values
(5) None of these

34. Which type of software manages the computers procesess, functioning as an interface, connecting the user, the application software and the hardware?
(1) System software
(2) Utility program
(3) Translator program
(4) Operating system
(5) None of these

SBI Clerk Exam, First Sitting, 13.07.2007

35. All of the following terms are associated with spread sheet software except.
(1) worksheet (2) cell
(3) formula
(4) virus detection
(5) None of these

36. _____ ,are words that a programming language has set aside for its own use.
(1) Control words
(2) Reserved words
(3) Reserved keys
(4) Control structures
(5) None of these

SBI Clerk Exam, Second Sitting, 13.07.2008

37. The primary purpose of software is to turn data into
(1) Web sites
(2) information
(3) programs (4) objects
(5) None of these

SBI Clerk Exam, 06.07.2008

38. _____ is the process of finding errors in software code.
(1) Compiling (2) Testing
(3) Running (4) Debugging
(5) None of these

SBI Clerk Exam, 06.07.2008

39. A _____ contains specific rules and words that express the logical steps of an algorithm.
(1) syntax
(2) programming structure
(3) programming language
(4) logic chart
(5) None of these

SBI Clerk Exam, 06.07.2008

40. A single application that combines the major features of several types of applications is called
(1) integrated software
(2) a suite
(3) a combo package
(4) high-end
(5) None of these

SBI PO Tier-I Exam, 27.07.2008

41. Application software
(1) is used to control the operating system
(2) is designed to help programmers
(3) performs specific task for computer users
(4) is used for making design only
(5) All of the above

42. A program embedded in a semiconductor chip during their manufacture is called
(1) humanware
(2) firmware (3) liveware
(4) hardware (5) software

SBI Clerk Exam, 10.02.2008

43. In the "C" language the function scanf ( ) reads
(1) single character
(2) character and strings
(3) any possible number
(4) any possible variable type
(5) limited variable types

44. An image editing software in which we can draw and edit images, is
(1) PageMaker (2) MS-Paint
(3) Photo Image
(4) Front page (5) Corel Draw

45. A set of instructions telling the computer what to do is called
(1) mentor (2) instructor
(3) compiler (4) program
(5) debugger
SBI Clerk Exam, 06.01.2008

46. Which of the following computer languages is a mathematically oriented languages used for scientific problems?
(1) FORTRAN
(2) COBOL (3) LISP
(4) PROLOG (5) APPLE
SBI Clerk Exam, 25.11.2007

47. What type of program controls the various computer parts and allows the user to interact with the computer ?
(1) Utility software
(2) Operating system
(3) Word processing software
(4) Database program
(5) None of these

48. What is the process of copying software programs from secondary storage media to the hard disk called ?
(1) configuration
(2) download (3) storage
(4) upload (5) installation
SBI Clerk Exam, 19.08.2007

49. How can the user determine what programs are available on a computer ?
(1) Checking the hard disk properties
(2) Viewing the installed programs during the booting process
(3) Checking the operating system for a list of installed programs
(4) Checking the existing files saved on the disk
(5) None of these
SBI Clerk Exam, 19.08.2007

50. The first computers were programmed using ——
(1) assembly language
(2) machine language
(3) source code
(4) object code
(5) spaghetti code
SBI Clerk Exam, 19.08.2007

ANSWERS

1. (1)	**2.** (2)	**3.** (4)	**4.** (4)
5. (2)	**6.** (3)	**7.** (1)	**8.** (2)
9. (2)	**10.** (3)	**11.** (1)	**12.** (1)
13. (4)	**14.** (1)	**15.** (2)	**16.** (1)
17. (3)	**18.** (4)	**19.** (3)	**20.** (2)
21. (1)	**22.** (1)	**23.** (4)	**24.** (1)
25. (1)	**26.** (1)	**27.** (1)	**28.** (3)
29. (5)	**30.** (3)	**31.** (3)	**32.** (4)
33. (2)	**34.** (4)	**35.** (4)	**36.** (4)
37. (2)	**38.** (2)	**39.** (1)	**40.** (1)
41. (3)	**42.** (2)	**43.** (4)	**44.** (5)
45. (4)	**46.** (1)	**47.** (1)	**48.** (3)
49. (4)	**50.** (2)		

EXPLANATIONS

6. (3) Compiler converts the source code (High Level Language) into machine code which can be run directly by the operating system as an executable program?
Interpreters bypass the compilation process and execute the code directly. Interpreters read and execute code in a single step; they are useful for running scripts and other small programs.

17. (3) An instruction prepared in machine language has two parts.
Op-code is the command or operation and it tells the computer what function to perform.
Operand tells the computer where to find or store the data or instructions that is to be manipulated.

19. (3) An assembler translates the symbolic instruction code of programs written in Assembly language into machine language.

20. (2) Assembly languages have the structure and set of commands and name which is easy to understand but machine language have numbers and command which not easy to understand.

22. (1) High level languages are sometimes classified as:
1. Problem oriented languages
2. Procedure oriented languages
3. Interactive programming languages

29. (5) C, C++, Java, COBOL are the computer programming languages while Microsoft is an application software.

38. (2) Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. It consists of various methods to test and declare a software product fit before it can be launched for use by either an individual or a group. It involves tests on functionality, performance and appearance. Correcting errors in a program is called Debugging.

42. (2) Most embedded devices contain firmware at more than one level. Subsystems such as CPUs, flash chips, communication controllers, LCD modules, and so on, have their own (usually fixed) program code and/or microcode, regarded as "part of the hardware" by the higher-level(s) firmware.

43. (4) Scanf is a function that reads formatted string, character, or numeric data which called variable from a given string stream source, originated from C programming language, and is present in many other programming languages.

44. (5) CorelDraw is a vector graphics editor. A vector graphics editor is a computer program that allows users to compose and edit vector graphics images interactively on a computer.

46. (1) FORTRAN is used for Scientific & Engineering while COBOL is orientated language Commercial and PROLOG is a general purpose logic programming language associated with artificial intelligence and computational linguistics