Q. 1 a) What is meant by computer software? Explain it in detail with the help of proper examples.

Answer:

Computer software, or simply software, is a part of a computer system that consists of data or computer instructions, in contrast to the physical hardware from which the system is built. In computer science and software engineering, computer software is all information processed by computer systems, programs and data. Computer software includes computer programs, libraries and related non-executable data, such as online documentation or digital media. Computer hardware and software require each other and neither can be realistically used on its own.

At the lowest level, executable code consists of machine language instructions specific to an individual processor—typically a central processing unit (CPU). A machine language consists of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. For example, an instruction may change the value stored in a particular storage location in the computer—an effect that is not directly observable to the user. An instruction may also (indirectly) cause something to appear on a display of the computer system—a state change which should be visible to the user. The processor carries out the instructions in the order they are provided, unless it is instructed to "jump" to a different instruction, or is interrupted (by now multi-core processors are dominant, where each core can run instructions in order; then, however, each application software runs only on one core by default, but some software has been made to run on many).

The majority of software is written in high-level programming languages that are easier and more efficient for programmers to use because they are closer than machine languages to natural languages. High-level languages are translated into machine language using a compiler or an interpreter or a combination of the two. Software may also be written in a low-level assembly language, which has strong correspondence to the computer's machine language instructions and is
Computer software is programming code executed on a computer processor. The code can be machine-level code, or code written for an operating system. An operating system is software intended to provide a predictable and dependable layer for other programmers to build other software on, which are known as applications. It also provides a dependable layer for hardware manufacturers. This standardization creates an efficient environment for programmers to create smaller programs, which can be run by millions of computers. Software can also be thought of as an expression that contrasts with hardware. The physical components of a computer are the hardware; the digital programs running on the hardware are the software. Software can also be updated or replaced much easier than hardware. Additionally, software can be distributed to a number of hardware receivers. Basically, software is the computer logic computer users interact with.

Two Basic Examples

A machine-level example of software is Basic Input/Output System, or BIOS. When you start the computer, the BIOS loads and runs before your hard drive even connects. The BIOS checks connection to hardware and looks for the operating system to load. You can upgrade the BIOS by flashing, which is when you replace machine-level software stored on the main board of your computer.

A familiar example of application software is Notepad. Notepad runs when the user activates it and it has certain requirements. You need an operating system and hardware processor. The programmers of Notepad wrote software for a specific environment. Once the software is loaded into the computer's memory, the processor is able to read it. The program then becomes a process, and the user can interact with it.

b) Differentiate between system software and application software.

Answer:

System Software vs Application Software

System software manages and operates computer hardware thereby providing a
platform for other application software. The one name that comes to mind in hearing the words “system software” is Operating System like Linux, Mac OS X, or Windows. Operating System makes it possible for different computer parts to work in tandem. It is also known as low-level software as it operates on the lowest computer level. It makes data movement possible between the memory and disks and manages the output to display devices. Another system software is BIOS and firmware. This helps in operating the built-in or connected computer hardware. We can make use of utilities like language translator, compilers, DBMS programs, and other diagnostic tools that are a special type of system software.

Application software is a subset of system software with the ability to employ certain computer functions directly as per user requirements. It gives the user the power to perform either single or multiple tasks. User specific software like animations, graphics, or macros follows under the aegis of application software. Other popular application software includes: CRM software, ERP software, accounting, graphics, and media software.

Many a times it becomes difficult to distinguish the subclass of application software from the main class of system software. But there are a few basic differences that help our understanding of the matter:

To understand the difference between system software and application software in layman terms, consider the example of a hydroelectric plant and tap water. Here “hydroelectric plant” is analogous to “system software” while “tap water” symbolizes “application software.”

In embedded systems like software used in microwave ovens and DVD players, it’s difficult to tread the thin line between application software and operating system software.

Well programmed system software abstracts its complex details from application software.

The number of system software running on your machine is less than application software as the presence of the latter depends on the user requirements.

System software can exist as a lone entity whereas application software needs system software for its existence.
Summary:

1. System software is an integral part that aids in the computer’s functioning. It manages the computer resources in ways that they can operate in tandem.

2. Application software is created for users. They manage their specific tasks to suit their needs like a media player or word processors.

3. System software provides a platform for the execution of application software.

Q. 2 What is meant by operating system? Define function of an operating system in detail with the help of suitable examples.

Answer:

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer—from cellular phones and video game consoles to web servers and supercomputers.

The dominant desktop operating system is Microsoft Windows with a market share of around 82.74%. macOS by Apple Inc. is in second place (13.23%), and the varieties of Linux are collectively in third place (1.57%). In the mobile (smartphone and tablet combined) sector, use in 2017 is up to 70% of Google's Android[4] and according to third quarter 2016 data, Android on smartphones is dominant with 87.5 percent and a growth rate 10.3 percent per year, followed by Apple's iOS with 12.1 percent and a per year decrease in market share of 5.2 percent, while other operating systems amount to just 0.3 percent. Linux
distributions are dominant in the server and supercomputing sectors. Other specialized classes of operating systems, such as embedded and real-time systems, exist for many applications.

An operating system has **three main functions:**

1. manage the computer's resources, such as the central processing unit, memory, disk drives, and printers,
2. establish a user interface, and
3. execute and provide services for applications software.

Keep in mind, however, that much of the work of an operating system is hidden from the user; many necessary tasks are performed behind the scenes. In particular, the first listed function, managing the computer's resources, is taken care of without the user being aware of the details. Furthermore, all input and output operations, although invoked by an applications program, are actually carried out by the operating system. Although much of the operating system functions are hidden from view, you will know when you are using an applications software package, and this requires that you invoke-call into action-the operating system. Thus you both establish a user interface and execute software.

Operating systems for mainframe and other large computers are even more complex because they must keep track of several programs from several users all running in the same time frame. Although some personal computer operating systems-most often found in business or learning environments-can support multiple programs and users, most are concerned only with a single user. We begin by focusing on the interaction between a single user and a personal computer operating system.

---

**Q. 3 a) Compare feature of Windows Operating System on your computer with other Operating System (use Internet).**

**Answer:**

Before any newly manufactured computer can be used, an operating system (OS) needs to be installed on it. All the components in the operating system serve to make the various parts of a computer function in synergy. Several operating
systems have been developed over time but the Windows OS and Mac OS are much more popular.

**Mac OS:** The Mac OS is basically run by the various models of computers manufactured by Apple Inc. These computers are known by the brand name Mac. Macs are able to run all the main operating systems available today and with certain software like Parallels Desktop; can run more than one OS simultaneously. The Mac OS runs on a UNIX base which makes it very stable, secure and easy to use. The displayed graphical interface is also considered user friendly by many people.

**PC (Windows) OS:** A computer that runs the Windows OS is generally referred to as a PC. Their design is based on IBM microprocessors. A PC cannot run all the operating systems out there, but the popular ones, such as Microsoft Windows and Linux, are compatible with it.

**Differences between Mac and PC**

**Cost:** Macs are much more expensive, with some models costing well over $1000. In contrast, PCs are cheaper and some are priced as low as 40% cheaper than comparable Macs.

**Manufacturer/Distributor:** Mac Os and Mac hardware are all exclusively produced, distributed and marketed by Apple Inc. There are however, several companies that make and distribute PCs and they include HP, Dell, Acer, Toshiba, Lenovo, Gateway, Samsung etc. This is a reason PCs cost less than Macs, as each company is likely to offer competitive prices to attract more buyers.

**Popular Applications:** The Mac OS has its own unique applications, which are iTunes, iPhoto, iBooks, iMovie, Pages, GarageBand, Time Machine, Photo Booth, FaceTime, Safari, Keynote, etc. The common PC applications are Internet Explorer, MS Office Suite, Windows Defender, Chrome Browser, Windows Media Center, VLC Media Player, SkyDrive, etc.

**Compatibility:** Macs can open most of PC file formats, such as .xls, .doc, .exe and others. Windows OS can seamlessly run on a Mac with no compatibility issues. This is not so with PCs. Without using software that can open Mac OS-based files (.DMG), a PC cannot read .DMG files.
**Performance:** Mac OS operates efficiently without lagging and has stable, high rated performance, as only Apple Inc handles all its OS and hardware updates. PCs often have incompatibility and lagging issues as they are produced by different companies and may not have the right drivers for specific OS and model updates. PC performance may be below expectation.

Strengths of Mac OS

1. Due to its UNIX core, it is very secure and much less vulnerable to destructive virus attacks.
2. Mac OS is very stable because all its parts are designed, manufactured and tested by one company.
3. The well designed graphic user interface (GUI) of Mac OS is very user friendly.
4. Other Apple products, such as iPad, and even non-Apple ones are recognized by Mac OS without having to install other drivers for them.
5. It can run most other operating systems and can even run Windows XP side by side using Parallels Software or Boot Camp.

**Weaknesses of Mac OS**

1. Macs and Mac OS are very expensive compared to other computers and operating systems.
2. Most computer video games are not run by Mac OS and only few games are made for it.
3. There are fewer software choices available here.

**Strengths of PC operating System**

1. A lot of people find it easier to use as the developers of the software have maintained its basic features from the earlier versions to the most recent updates.
2. A large selection of applications and software catering to different purposes have been made for PC.
3. There is backward compatibility, as the older versions can still work with more
current Windows versions.

4. New hardware is easily recognized by this OS.

5. Many video games are designed to be compatible with PC.

**Weaknesses of PC operating system**

1. Its security is weak and easier for hackers to penetrate.

2. There is high vulnerability to viruses and other malware.

3. Technical support service is mostly inadequate.

4. The system is comparatively unstable.

5. Older hardware gets poor continuous support.

After thoroughly comparing these two operating systems, the potential user can then choose which best suits them depending on the programs to be executed, hardware features desired and the available budget. Mac hardware and OS are available at Apple outlets and there are various models to choose from. The popular choices are either the Mac mini desktop, Mac-book Air notebook or the iMac. PC hardware and OS are readily available in a range of models as there are more companies designing and producing them.

b) **Briefly explain the terms: assembler, compiler, linker, and interpreter.**

**Answer:**

An **assembly** (or assembler) language, often abbreviated asm, is a low-level programming language for a computer, or other programmable device, in which there is a very strong (but often not one-to-one) correspondence between the language and the architecture's machine code instructions. Each assembly language is specific to a particular computer architecture. In contrast, most high-level programming languages are generally portable across multiple architectures but require interpreting or compiling. Assembly language may also be called symbolic machine code. Assembly language is converted into executable machine code by a utility program referred to as an assembler. The conversion process is
referred to as assembly, or assembling the source code. Assembly time is the computational step where an assembler is run.

A **compiler** is a program that translates the source code for another program from a programming language into executable code. The source code is typically in a high-level programming language (e.g. Pascal, C, C++, Java, Perl, C#, etc.). The executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine (e.g. Java byte code).

In short, a compiler converts a program from a human-readable format into a machine-readable format. As to how a compiler works, that is indeed complicated. There are books and university courses on the subject. I will attempt to briefly outline the main stages of the process, but this will be a very cursory overview.

In computing, a **linker** or link editor is a computer program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another 'object' file. A simpler version that writes its output directly to memory is called the loader, though loading is typically considered a separate process. Computer programs typically are composed of several parts or modules; these parts/modules need not all be contained within a single object file, and in such cases refer to each other by means of symbols. Typically, an object file can contain three kinds of symbols:

- defined "external" symbols, sometimes called "public" or "entry" symbols, which allow it to be called by other modules,
- undefined "external" symbols, which reference other modules where these symbols are defined, and
- local symbols, used internally within the object file to facilitate relocation.

For most compilers, each object file is the result of compiling one input source code file. When a program comprises multiple object files, the linker combines these files into a unified executable program, resolving the symbols as it goes along.

Linkers can take objects from a collection called a **library**. Some linkers do not include the whole library in the output; they include only its symbols that are
referred from other object files or libraries. Libraries exist for diverse purposes, and one or more system libraries are usually linked in by default.

In computer science, an **interpreter** is a computer program that directly executes, i.e. *performs*, instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program. An interpreter generally uses one of the following strategies for program execution:

- parse the source code and perform its behavior directly;

- translate source code into some efficient intermediate representation and immediately execute this;

- explicitly execute stored precompiled code made by a compiler which is part of the interpreter system.

Early versions of Lisp programming language and Dartmouth BASIC would be examples of the first type. Perl, Python, MATLAB, and Ruby are examples of the second, while UCSD Pascal is an example of the third type. Source programs are compiled ahead of time and stored as machine independent code, which is then linked at run-time and executed by an interpreter and/or compiler (for JIT systems). Some systems, such as Smalltalk and contemporary versions of BASIC and Java may also combine two and three. Interpreters of various types have also been constructed for many languages traditionally associated with compilation, such as Algol, Fortran, Cobol and C/C++

{================}

Q. 4 a) **Explain the basic concept of multimedia with the help of proper examples.**

**Answer:**

Multimedia is content that uses a combination of different content forms such as text, audio, images, animations, video and interactive content. Multimedia contrasts with media that use only rudimentary computer displays such as text-only or traditional forms of printed or hand-produced material.

Multimedia can be recorded and played, displayed, interacted with or accessed by information content processing devices, such as computerized and electronic
devices, but can also be part of a live performance. Multimedia devices are electronic media devices used to store and experience multimedia content. Multimedia is distinguished from mixed media in fine art; for example, by including audio it has a broader scope. In the early years of multimedia the term "rich media" was synonymous with interactive multimedia, and "hypermedia" was a application of multimedia.

**Major characteristics**

Multimedia presentations may be viewed by person on stage, projected, transmitted, or played locally with a media player. A broadcast may be a live or recorded multimedia presentation. Broadcasts and recordings can be either analog or digital electronic media technology. Digital online multimedia may be downloaded or streamed. Streaming multimedia may be live or on-demand.

Multimedia games and simulations may be used in a physical environment with special effects, with multiple users in an online network, or locally with an offline computer, game system, or simulator.

The various formats of technological or digital multimedia may be intended to enhance the users' experience, for example to make it easier and faster to convey information. Or in entertainment or art, to transcend everyday experience.

Enhanced levels of interactivity are made possible by combining multiple forms of media content. Online multimedia is increasingly becoming object-oriented and data-driven, enabling applications with collaborative end-user innovation and personalization on multiple forms of content over time. Examples of these range from multiple forms of content on Web sites like photo galleries with both images (pictures) and title (text) user-updated, to simulations whose coefficients, events, illustrations, animations or videos are modifiable, allowing the multimedia "experience" to be altered without reprogramming. In addition to seeing and hearing, haptic technology enables virtual objects to be felt. Emerging technology involving illusions of taste and smell may also enhance the multimedia experience.

{================}
b) Identify at least ten different examples of multimedia and also describe five of them in detail.

Answer:

Multimedia presentations may be viewed by person on stage, projected, transmitted, or played locally with a media player. A broadcast may be a live or recorded multimedia presentation. Broadcasts and recordings can be either analog or digital electronic media technology. Digital online multimedia may be downloaded or streamed. Streaming multimedia may be live or on-demand.

Multimedia games and simulations may be used in a physical environment with special effects, with multiple users in an online network, or locally with an offline computer, game system, or simulator. The various formats of technological or digital multimedia may be intended to enhance the users' experience, for example to make it easier and faster to convey information. Or in entertainment or art, to transcend everyday experience.

**Five different examples of multimedia:**

1. **Multimedia Portfolio**

When you need to quickly show off a vast array of what you can accomplish with the PDF, there's no easier way than in a PDF portfolio. They're a great way to combine a number of multimedia elements into a single format.

2. **Video Elements**

Movement attracts the human eye, so why not attract more eyes with videos embedded directly into your PDF files. Here's a great PDF that puts the embedded video element center stage.

3. **3D Elements**

Make that information leap off the page with 3D elements and highlights in your PDFs. With the right tools and some patience, you can even make them interactive.

4. **Interactive Flash**

Flash can be implemented through a variety of ways, though many of them will require another application or tool to help create the Flash items. Adding a Flash
movie is just like adding a video (see above). Take a look at these e-brochures from

5. Audio Elements
Some information is best transferred via sound. So why not hook some audio into your PDF? Find out how to effectively integrate sound with your content in this e-brochure from Uniquely Merit that showcases group vacations to a number of different countries. Cultural music specific to each country plays in the background as you turn through the pages.

Q. 5 Write short notes (in your own words) on the following topics:

Computer Languages

Answer:
A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, Java, FORTRAN, Ada, and Pascal. Each programming language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

**High-Level Programming Languages**
High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages. Each different type of CPU has its own unique machine language.

Lying between machine languages and high-level languages are languages called assembly languages. Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers. Machine languages consist of numbers only.

Lying above high-level languages are languages called fourth-generation languages (usually abbreviated 4GL). 4GLs are far removed from machine languages and represent the class of computer languages closest to human
languages.

**Converting to Machine Language**

Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it. There are two ways to do this:

1) Compile the program.

2) Interpret the program.

The question of which language is best is one that consumes a lot of time and energy among computer professionals. Every language has its strengths and weaknesses. For example, FORTRAN is a particularly good language for processing numerical data, but it does not lend itself very well to organizing large programs. Pascal is very good for writing well-structured and readable programs, but it is not as flexible as the C programming language. C++ embodies powerful object-oriented features, but it is complex and difficult to learn.

```
Development of Software

Answer:

Software development is the process of computer programming, documenting, testing, and bug fixing involved in creating and maintaining applications and frameworks resulting in a software product. Software development is a process of writing and maintaining the source code, but in a broader sense, it includes all that is involved between the conception of the desired software through to the final manifestation of the software, sometimes in a planned and structured process. Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a perceived need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a scientist may
write software to automate a mundane task). Embedded software development, that is, the development of embedded software, such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product. System software underlies applications and the programming process itself, and is often developed separately.

The need for better quality control of the software development process has given rise to the discipline of software engineering, which aims to apply the systematic approach exemplified in the engineering paradigm to the process of software development.

There are many approaches to software project management, known as software development life cycle models, methodologies, processes, or models. The waterfall model is a traditional version, contrasted with the more recent innovation of agile software development.

A software development process (also known as a software development methodology, model, or life cycle) is a framework that is used to structure, plan, and control the process of developing information systems. A wide variety of such frameworks has evolved over the years, each with its own recognized strengths and weaknesses. There are several different approaches to software development: some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations.