Software

(Chapter 4)

Syllabus Content:

4.1 Types of software and interrupts

Candidates should be able to:	Notes and guidance
1 Describe the difference between system software and application software and provide examples of each	• System software provides the services that the computer requires, including operating system and utility software • Application software provides the services that the user requires
2 Describe the role and basic functions of an operating system	• Including: – managing files – handling interrupts – providing an interface – managing peripherals and drivers – managing memory – managing multitasking – providing a platform for running applications – providing system security – managing user accounts

4.1 Types of software and interrupts continued

Candidates should be able to:	Notes and guidance
3 Understand how hardware, firmware and an operating system are required to run applications software	• Applications are run on the operating system • The operating system is run on the firmware • The bootloader (firmware) is run on the hardware
4 Describe the role and operation of interrupts	• Including: – how an interrupt is generated – how it is handled using an interrupt service routine – what happens as a result of the interrupts • Software interrupts include division by zero and two processes trying to access the same memory location • Hardware interrupts include pressing a key on the keyboard and moving the mouse

Syllabus Content:

4.2 Types of programming language, translators and integrated development environments (IDEs)

Candidates should be able to:	Notes and guidance
1 Explain what is meant by a high-level language and a low-level language, including the advantages and disadvantages of each	• Advantages and disadvantages include: – ease of reading and writing code, e.g. low-level is hard to read – ease of debugging code – machine independence – direct manipulation of hardware
2 Understand that assembly language is a form of low-level language that uses mnemonics, and that an assembler is needed to translate an assembly language program into machine code	
3 Describe the operation of a compiler and an interpreter, including how high-level language is translated by each and how errors are reported	• A compiler translates the whole code at once before executing it, producing an executable file • An interpreter translates and executes the code line-by-line • A compiler provides an error report for the whole code if errors are detected • An interpreter stops execution when an error is found

4.2 Types of programming language, translators and integrated development environments (IDEs) continued

Candidates should be able to:	Notes and guidance
4 Explain the advantages and disadvantages of a compiler and an interpreter	• To include an understanding that an interpreter is mostly used when developing a program and a compiler is used to translate the final program
5 Explain the role of an IDE in writing program code and the common functions IDEs provide	• Including: – code editors – run-time environment – translators – error diagnostics – auto-completion – auto-correction – prettyprint

4 Software

4.1 | Types of Software & Interrupts

4.1.1 System Software & Application Software:

NOTE: System Software & Application Software are newly added topics in the Computer Science (2210) syllabus for the session 2023–2025.

There are two types of software: system software & application software

System Software:

- The system software provides the services that the computer requires, including operating system and utility software.
- It controls and manages computers hardware and runs the applications software.
- It provides the interface between the computer hardware and the user applications.

Examples of system software include:

1. Operating system (OS)
2. Utility programs
3. Device drivers
4. Compilers
5. File management system
6. Linkers

Application Software:

- The application software provides the services that the user requires.
- It allows the user to carry out specific tasks such as creating a document or an image.

Examples of application software include:

1. Spreadsheet
2. Word processor
3. Database
4. Internet browser
5. Apps
6. Games software
7. Photo editing software

General features of system software & application software:

System Software	Application Software
These are set of programs that control and manage operation of computer hardware	It is used to perform various applications (apps) on a computer
It provides a platform on which other software can run	It allows a user to perform specific tasks using the computers resource
It is required to allow hardware and software to run without problems	It may be a single program (e.g. Notepad) or a suite of programs (e.g. Microsoft Office)
It provides a human computer interface (HCI)	The users can run the software whenever they want

Examples of typical Application Software:

1) Word Processor: It is used to manipulate a text document.

Functions:

1. Creating, editing, saving & manipulating text
2. Copy and paste functions
3. Spell checkers and thesaurus
4. Import photos/images into a structured page format

2) Spreadsheet: It is used to organize and manipulate numerical data.

Functions:

1. Use of formulas to carry out calculations
2. Ability to produce graphs
3. Ability to do modelling and "what if" calculations

3) Database: It is used to organize, manipulate and analyze data.

Functions:

1. Ability to carry out queries on database data and produce a report
2. Add, delete and modify data in a table

4) Control & Measuring Software:

It allows a computer or microprocessor to communicate with sensors for following purposes:

1. Measure physical quantities in real world (e.g. temperature)
2. To control applications such as a chemical process by sending signals

5) Applications (Apps): It is normally a software which runs on mobile phones or tablets (e.g. games, phone banking app etc.).

Examples of Apps:

1. Video and music streaming (e.g. YouTube, Spotify etc.)
2. GPS (global positioning systems)
3. Camera facility
4. Social networking apps (e.g. Instagram, Facebook, Twitter etc.)
5. Instant messaging apps (e.g. WhatsApp)
6. Games (e.g. Subway Surfer)

6) Photo Editing Software: It allows manipulation of digital photographs.

Functions:

1. Resize; increase/decrease size of the image
2. Crop; remove part of the image
3. Blur; reduce the focus
4. Red eye reduction; reduces red light reflected from human eyes

7) Sound Editing Software: It allows manipulation of digital sound.

Functions:

1. Fading; change the volume of a section of the sound for it get louder/quieter
2. Removing sound/elements; delete sections of sound wave (e.g. background noise)
3. Copy; repeat elements of the sound wave
4. Cut/delete; remove part of the sound file
5. Amplify; increase the volume of a section of sound

8) Video Editing Software: It allows manipulation of digital videos to produce a new video.

Functions:

1. Trim; adding and/or removing sections of video clips
2. Applying color corrections, filters and other video enhancements
3. Creating transitions between clips in the video footage
4. Slow motion feature

9) Graphics Manipulation Software: It allows bitmaps and vector images to be changed.

Functions:

1. Pixels of bitmap images can be changed to produce a different image
2. The lines & curves of vector images can be manipulated to change the stored image

Examples of typical System Software:

1) Operating System (OS): It provides an interface between the user and the hardware.

Functions:

1. Providing an interface
2. Managing memory
3. Providing system security
4. Managing peripherals and drivers
5. Managing files
6. Handling interrupts
7. Providing a platform for running applications
8. Managing multitasking
9. Managing user accounts

2) Device Drivers: It is a software that enables one or more hardware devices to communicate with the computers operating system (OS). All hardware devices connected to a computer have associated drivers.

Examples of Drivers:

1. Printers
2. Memory sticks
3. Mouse
4. CD drivers

3) Compilers: It is a computer program that translates a program written in a high-level language (HLL) into machine code.

Functions:

1. It translates the whole program (the source code) as a complete unit/all at once/all in one go.
2. It creates an executable file/object code.
3. A report/list of errors in the code is created.
4. It optimizes the source code to run efficiently.

4) Linkers (link editor): It is a computer program that takes one or more object file produced by a compiler and combines them into a single program which can be run on a computer.

Functions:

1. The programming languages allow programmers to write different pieces of code which are separately called 'modules'.
2. It simplifies programming task since program is broken up into small, more manageable sub-tasks.
3. Linker puts all modules together to form the final program.

5) Utility Software (utilities):

- They are a part of the operating system of a computer.
- These are additional programs that help to maintain or configure the system.
- They are routines which carry out important tasks which are necessary.

Examples of Utility Software are:

1. Virus Checker (Anti-Virus Software)
2. Defragmentation Software
3. Disk Contents Analysis/Repair Software
4. Disk Cleanup/System Cleanup Software
5. Disk Formatter Software
6. Disk Mirroring Software
7. File Compression Software
8. Back-up Software
9. Security Software

The following table shows whether the given examples are of applications software or system software:

Software	Applications (✓)	System (✓)
Control software	✓	
Measurement software	✓	
Compiler		✓
Word processing software	✓	
Device drivers		✓
Linker		✓
Database management systems	✓	
Database	✓	
Photo-editing software	✓	
Compiler		✓
Spreadsheet software	✓	
Operating system		✓
Applet	✓	
Utility software		✓

Utility Softwares:

1) Virus Checker (Anti-Virus Software):

- It scans files stored on a computer system for malicious code.
- It scans files when they enter the system e.g., memory stick inserted, or file downloaded etc.
- It sets up a schedule for virus-checking.
- It isolates/quarantines/deletes viruses.
- It makes more RAM available for programs to run because it removes software that might be taking up memory/replicating.

When does the virus checker perform a check?

- It checks for boot sector viruses when the machine is first turned on.
- It checks for viruses when an external storage device is connected e.g., memory stick.
- It checks for viruses when a file/web page is accessed/downloaded.

2) Defragmentation Software:

- It finds files that are split across the disk.
- It moves individual files together, so they occupy contiguous blocks (blocks which are next to each other/closer to each other).
- This creates a larger area of (contiguous) free space.
- It re-organizes the disk contents.
- It improves disk access times as data/files can be loaded faster because each file is contiguous (closer) to each other so there is less head movement.

3) Disk Contents Analysis/Repair Software:

- It checks for any errors/bad sectors on the disk as some areas of disk can become corrupt.
- It resolves any errors on the disk.
- It marks bad sectors on the disk as unusable, which reduces access time by optimizing storage.
- It recovers damaged files/data from a damaged disk (recovers disc when data gets corrupt).

4) Disk Cleanup/System Cleanup Software:

- It releases storage by removing unwanted/temporary files.
- This improves system performance as memory is saved from unwanted data.

5) Disk Formatter Software:

- It creates logical drives (partitions) on a hard drive.
- It reformats a previously used hard drive.
- It prepares a disk for first use.
- It sets up the specified file system.
- It may check for errors on the disk.

6) Disk Mirroring Software:

- The data is stored on two disks simultaneously.
- If the first disk drive fails, the data is accessed from the second disk.

7) File Compression Software:

- It reduces the file size by using algorithms to change the data which can be either lossy or lossless compression.

8) Back-up Software:

- It makes a copy of data at regular intervals (e.g., daily, weekly, monthly etc.).
- The copy of data is stored in a different location/elsewhere.
- If the original data is lost/corrupted, then it can be restored from the backup.
- It allows the user to decide what is backed up (e.g., all data // all files that have changed since the last backup).
- It may encrypt the backup files.
- It creates a restore point where your computer can be restored to its state at this earlier point in time.

9) Security Software:

- It manages access control and user accounts (using user IDs and passwords).
- It links into other utility software, such as virus checkers and spyware checkers.
- It protects network interfaces e.g., through the use of firewalls.
- It uses encryption and decryption to ensure any intercepted data is meaningless without a decryption key.
- It checks the updating of software e.g., update request comes from a legitimate source.

How the disk formatter, disk contents analysis and disk repair utilities work together.

- The disk contents analysis checks for errors/problems with the disk.
- The disk repair attempts to fix the errors.
- The disk formatter prepares the disk for initial use again.

The following table shows whether the given programs are utility software or not:

Program	Utility (✓)	Not utility (✓)
Disk Defragmenter	✓	
Word Processor		✓
Library Program		✓
Compression Software	✓	
Database		✓
Virus Checker	✓	
Web Browser		✓
Backup Software	✓	
Language Translator		✓
Integrated Development Environment (IDE)		✓
Graphics Software		✓
Spreadsheet		✓

A few examples are given below which will help you better understand the utility software solutions required for certain scenarios/statements according to the examination question.

Example 1:

Statement	Utility software solution
The hard disk stores a large number of video files. The computer frequently runs out of storage space.	File compression software
The user is unable to find an important document. He thinks it was deleted in error some weeks ago. This must not happen again.	Backup software
The operating system reports 'Bad sector' errors.	Disk repair software
There have been some unexplained images and advertisements appearing on the screen. The user suspects it is malware.	Anti-virus software

Example 2:

Statement	Utility software solution
The user wants to send a large file as an attachment to an email. The user knows that the recipient's Internet Service Provider (ISP) has a limit of 2MB for file attachments.	File compression software
The user is writing a book and is worried that the document files could get damaged or deleted.	Backup software
The computer has recently been slow to load large files. The user has deleted a large number of small files to try to solve the problem. A friend has advised that there is a procedure which should be regularly carried out to reorganize file storage on the hard disk.	Disk defragmenting software
The user clicked on an attachment in an unsolicited email. Since then, the computer has shown some unexplained behaviors.	Anti-virus software

Example 3:

Sam is a photographer. She has an image library of over 10 000 images. She stores the images on a high capacity magnetic hard disk.

Why Sam would use the following utility software:

(i) Backup software:

- It would make a copy of her files (which can be stored elsewhere).
- The images on the hard disk can be lost or corrupted.
- This allows for images to be restored.

(ii) Defragmenter software:

- The frequent changes to the images mean the data for each file is split across the disk.
- It finds those files and rearranges them into contiguous blocks.
- This creates more contiguous free space.
- It improves the time taken to access and load the files.

(iii) Disk repair software:

- It checks for any errors/bad sectors on the disk as some areas of disk can become corrupt.
- It marks the errors/bad sectors on the disk as unusable so they can be repaired.
- Those bad sectors are no longer used, which reduces access time.

Exam Style Questions:

Question 1:

A small company produces scientific magazines. The owner buys some new desktop computers. The computers are used to store thousands of colour images (diagrams and photographs). All the computers have Internet access.

(a) Name **three** utility programs the company would use on all their computers. Describe what each program does.

1 .....

Description .....

.....

2 .....

Description .....

.....

3 .....

Description .....

.....[6]

Answer:

(a) **One mark** for the name and **one mark** for the explanation for **three** utility programs

- Disk formatter
- Prepares a hard disk to allow data to be stored on it

- Virus checker
- Checks for viruses and then quarantines removes any virus found

- File compression
- Reduces file size by removing redundant details (lossy / lossless)

- Backup software
- Makes copy of files on another medium in case of corruption / loss of data

- Firewall
- Prevents unauthorised access to computer system from external sources [6]

Question 2:

(b) A hard disk formatter and a hard disk defragmenter are two examples of utility software.

(i) Describe the actions performed by a hard disk formatter and a hard disk defragmenter.

Hard disk formatter .....

.....

.....

.....

Hard disk defragmenter .....

.....

.....

.....

[4]

(ii) Identify **three other** examples of utility software that can be installed on the computer.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Answer:

1(b)(i)	**1 mark** per bullet point to **max 2** for formatter, **max 2** for defragmenter **hard disk formatter** • Makes existing data inaccessible • Partitions the disk into logical drives • Sets up the (specified) file system • Prepares the disk for initial use • May check for errors on the disk **hard disk defragmenter** • Re-organises the disk contents • Moves split files so they are contiguous • Creates a larger area of (contiguous) free space	**4**
1(b)(ii)	**1 mark** per bullet point For example: • Backup software • File compression • Virus checker • Disk contents analysis / repair	**3**

Question 3:

Utility programs are examples of system software.

(a) Complete the table by writing the name of the utility program for each description.

Description	Utility program
Reorganises files on a disk to improve efficiency	
Scans a hard disk to identify bad sectors	
Prepares a hard disk for first use	

[3]

Answer:

9(a)	**1 mark** for each correctly identified utility program **Description** / **Utility program** Reorganises files on a disk to improve efficiency — **Defragmentation software** Scans a hard disk to identify bad sectors — **Disk contents analysis / repair software** Prepares a hard disk for first use — **Disk formatter**	**3**

Question 4:

Identify the **most appropriate** utility program to use for each of the following tasks.

Task	Utility program
Rearrange the data on a disk so that files are contiguous, and all free space is collected together	
Prepare a disk for initial use	
Reduce the size of a file	
Examine a disk to find any bad sectors	

[4]

Answer:

1	**1 mark** for each correct utility program **Task** / **Utility program** Rearrange the data on a disk so that files are contiguous, and all free space is collected together — **Disk defragmenter** Prepare a disk for initial use — **Disk formatter** Reduce the size of a file — **File compression** Examine the disk to find any bad sectors — **Disk contents analysis / repair**	**4**

Question 5:

(c) Identify **two** utility programs that can be used to improve the performance of a computer **and** state how they improve the performance.

1 .....

.....

.....

.....

2 .....

.....

.....

.....

[4]

Answer:

7(c)	**1 mark** for identifying program **1 mark** for description, **max 2** per program e.g. • Defragmentation • Less time is taken to access files because each one is contiguous so there is less head movement • Virus checker • makes more RAM available for programs to run • ... because it removes software that might be taking up memory / replicating • Disk repair / Disk contents analysis • preventing bad sectors being used because it identifies / marks them • reduces access times by optimising storage • Disk/system clean up • releases storage by removing unwanted / temporary files	**4**

Question 6:

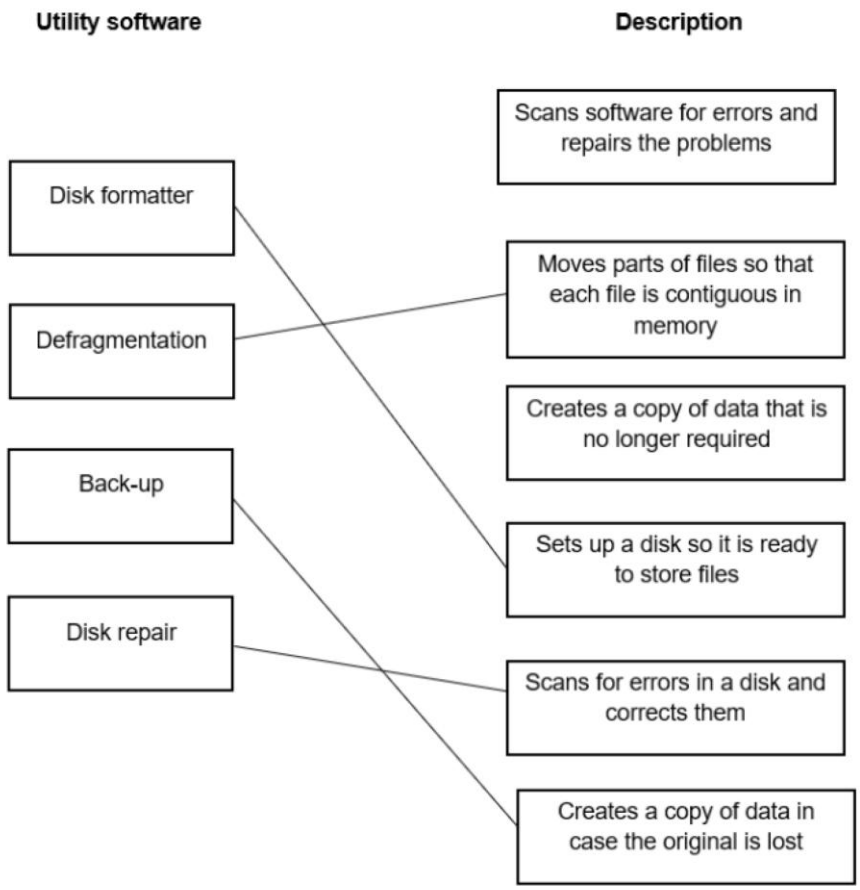
Bingwen's computer comes with an Operating System and utility software.

(a) Draw **one** line from each utility software to its correct description.

Utility software	Description
	Scans software for errors and repairs the problems
Disk formatter	Moves parts of files so that each file is contiguous in memory
Defragmentation	Creates a copy of data that is no longer required
Back-up	Sets up a disk so it is ready to store files
Disk repair	Scans for errors in a disk and corrects them
	Creates a copy of data in case the original is lost

[4]

Answer:

2(a)	**1 mark** for each correct line  	**4**

Question 7:

(d) There are two types of software: system and applications.

Give **two** examples of applications software.

1 .....

.....

2 .....

.....

[2]

Answer:

1(d)	**Two** from for example: Word processing Spreadsheet Database Presentation	**2**

Question 8:

(b) There are two types of software: system and applications.

Give **two** examples of system software.

1 .....

.....

2 .....

.....

[2]

Answer:

1(b)	**Two** from: Compilers Linkers Device drivers Operating systems Utilities	**2**

Question 9:

(a) Explain what is meant by system software.

.....

..... [1]

(b) Explain what is meant by applications software.

.....

..... [1]

(c) Circle **two** types of system software.

Actuator	Linker	Operating system
Photo-editing software	Sensor	Spreadsheet

[2]

Answer:

Question	Answer	Marks
5(a)	**One** from: Programs that control and manage the computer's hardware//it runs the applications software Interface between the computer hardware and the user applications	**1**
5(b)	Programs that allow the user to carry out specific tasks	**1**
5(c)	Linker Operating system	**2**

4.1.2 Operating System (OS):

- It provides an interface between the user and the hardware.

Why a Personal Computer (PC) needs an Operating System:

- It performs a number of basic tasks, including controlling hardware.
- It allows the user to communicate with the computer using hardware.
- It provides the user with a user interface.
- The PCs are often used to perform many complex tasks at a time and the OS is needed to handle this multitasking.
- Therefore, it provides the ability to handle interrupts.

Common examples of OS:

1. Microsoft Windows
2. Apple Mac OS
3. Google Android
4. Apples iOS

Basic functions of an Operating System (key management tasks):

1. Providing an interface
2. Managing memory
3. Providing system security
4. Managing peripherals and drivers
5. Managing files
6. Handling interrupts
7. Providing a platform for running applications
8. Managing multitasking
9. Managing user accounts

Functions/Tasks of Operating System:

1) Providing an interface (provision of a user interface):

- The operating system provides the Human Computer Interface (HCI) which allows the user to communicate with the hardware.
- It allows user interaction with the hardware/computer system by making navigation around the system easier.
- It provides a facility for user inputting data as well as outputting data to the user.
- It hides the complexity of the hardware from the user.

For example: command line interface / GUI / menu-driven

The Human Computer Interface (HCI) is in the form of:

1. Command Line Interface (CLI)
2. Graphical User Interface (GUI)

1) Command Line Interface (CLI):

- CLI (command line interface) is the one where user types a series of commands using keyboard and communicates directly with computer system.
- To use this, the user needs to know what commands are available, understand the commands and understand the way information is stored in a computer system.
- It is used by a programmer, analyst, or a technician (those who want direct communication with computer).

How a user can select and open an application using CLI including any input hardware needed:

- The user communicates directly with the computer system by typing in commands in response to a prompt.
- Several commands are entered to carry out a task such as simply loading software.
- The input hardware needed for typing these commands is keyboard or keypad.

Advantages of CLI:

1. It allows direct communication with the computer system.
2. It is not restricted to a number of pre-determined options.
3. It is a simple interface using keyboard only.
4. It uses a small amount of computer memory

Disadvantages of CLI:

1. A user needs to learn a number of long and complex commands.
2. A user needs to learn how information is stored on a computer.
3. The user needs to type in commands which results in possibility of errors.
4. It is slower to type in commands every time.

2) Graphical User Interface (GUI):

- Graphical interfaces are called GUI (graphical user interface) or WIMP (windows, icons, menus and pointer) because it displays graphics as well as text.
- GUI is very user friendly as the user can select options by use of menus of choices and small pictures/icons which represent different available options.
- Choices (icons) are selected by the user using a pointing device e.g., mouse and the selected application is opened/run.
- It is used by end-user who don't have or need to have any great knowledge of how computer works (a person who uses computer to run software, play games etc.).

How a user can select and open an application using GUI including any input hardware needed:

- The user interacts with a computer using pictures and symbols or a drop-down menu.
- The tasks are initiated by selecting the icons (e.g., selecting the icon of a software to open it).
- It is usually part of a windows environment.
- The input hardware needed for selections is a pointing device (e.g., mouse) or a touch screen.

Advantages of GUI:

1. The user only needs to click on one simple picture rather than learning commands.
2. It is more user-friendly.
3. It is much easier for the beginners as only a pointing device is used to click on an icon to launch the application.
4. Several instructions are replaced by just one icon.
5. There is no need to understand how a computer works.

Disadvantages of GUI:

1. It uses a considerable amount of computer memory.
2. The user is limited to the icons provided on the screen.
3. If the user wants direct communication, it is effectively more complex.

2) Managing memory (memory management):

- It controls the movement of data between RAM, processor, VM etc.
- It keeps track of allocated and free memory locations.
- It carries out memory protection to ensure that two programs do not try to use the same space.
- It organizes memory e.g., paging/segmentation.
- It makes use of virtual memory.
- It allocates memory to processes and decides which processes need to be in main memory at any one time and releases memory when a process stops.

For example: when the process terminates, memory is made available.

3) Providing system security (security management):

- It makes provision for recovery when data is lost.
- It ensures privacy of data.
- It sets up user accounts (provides usernames and passwords).
- It prevents unauthorized access.
- It implements access rights for all users.
- It provides & upgrades firewall/anti-virus software.
- It offers the ability for system restore (to previous stable state).

4) Managing peripherals and drivers (hardware peripheral management):

It involves all input and output peripheral devices therefore it is also sometimes referred to as:

1. Peripheral management
2. Hardware management
3. Device management
4. Input/Output management

All of these management tasks are the same:

- It manages communication between devices/hardware and software using device drivers.
- It controls access to data being sent to/from hardware/peripherals.
- It receives data from input devices and sends data to output device.
- It installs appropriate driver softwares.
- It keeps track of device status (free or busy) and ensures that every hardware resource has a priority.
- It manages interrupts/signals from the device.
- It manages/controls queues and buffers (buffer management)

Printer management is an example of queue & buffer management. The following are printer management tasks that the Operating System performs:

- It installs printer driver.
- It sends data to the printer buffer ready for printing.
- If a printer is busy, then it sends data to the print queue.
- It sends various control commands to the printer.
- It receives and handles error messages and interrupts from the printer.

5) Managing files (file management):

- It provides file naming conventions which can be used.
- It maintains/creates file directory structures and specifies the logical method of file storage (e.g. FAT or NTFS).
- It divides storage space into file allocation units and the space is allocated to particular files.
- It specifies tasks that can be performed on a file/folders (e.g. open, close, delete, copy, create, rename, move etc.)
- It ensures access control/implements access rights (implements password protection & makes file sharing possible)

6) Handling interrupts:

- It identifies priorities of interrupts.
- It loads appropriate Interrupt Service Routine (ISR).
- It halts the execution of the current process.
- It stores the values of the current process on the stack.
- It saves current memory/data on power outage.

7) Providing a platform for running applications:

- It provides an environment/software platform on which programs/applications can be run.

8) Managing multitasking (task management):

- It allocates resources to a process for a specific time limit.
- It decides which process to run next.
- The process can be interrupted while it is running.
- The processes are given a priority to ensure fair access.
- It prevents interference between processes // resolution of conflicts.

9) Managing user accounts:

- It oversees and maintains accounts for several users.
- It manages private user data for several people.
- It allows multi-access levels for better control of personal data.

10) Error detection and recovery management tasks:

- It deals with interrupts.
- It deals with runtime errors generated by software.
- It deals with hardware faults.
- It displays error diagnostic messages.
- It allows safe-mode boot-up routines.
- It saves system restore points.

Exam Style Questions:

Question 1:

State **four** functions of an operating system.

1 .....

2 .....

3 .....

4 .....

[4]

Answer:

Any **four** from:

- Provides a user interface
- Handles interrupts / errors
- Memory management
- File management
- Manages peripherals (inputs/outputs)
- Provides security methods
- Allows multitasking
- Manages multiprogramming
- Enables batch processing
- Manages software installation / removal
- Allows creation of multiple accounts
- Levels of access

[4]

Question 2:

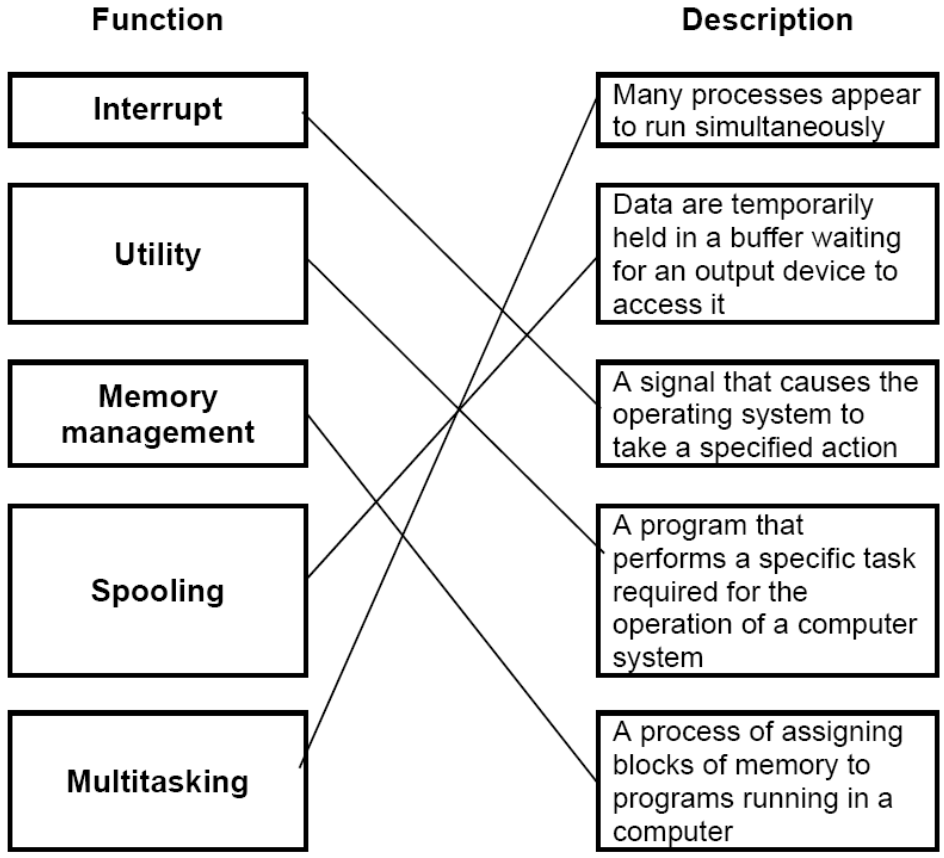
The diagram shows **five** operating system functions and **five** descriptions.

Draw a line between each operating system function and its description.

Function	Description
Interrupt	Many processes appear to run simultaneously
Utility	Data are temporarily held in a buffer waiting for an output device to access it
Memory management	A signal that causes the operating system to take a specified action
Spooling	A program that performs a specific task required for the operation of a computer system
Multitasking	A process of assigning blocks of memory to programs running in a computer

[4]

Answer:

6	1 mark for each correctly drawn line from a function to its description to a maximum of 4  	**4**

Question 3:

An operating system (OS) is usually pre-installed on a new computer.

(a) The OS performs a number of different tasks such as memory management and security management.

(i) State **three** memory management tasks the OS performs.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(ii) State **three** security management tasks the OS performs.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(iii) State **two** tasks, other than memory management and security management that are carried out by an OS.

1 .....

.....

2 .....

.....

[2]

Answer:

3(a)(i)	**1 mark** per bullet to **max 3** • Allocates / deallocates RAM to programs/tasks/processes • Keeps track of allocated and free memory locations • Swaps data to and from the hard drive • Handles virtual memory • Paging // segmentation • Memory protection, preventing a process accessing memory not allocated to it	**3**
3(a)(ii)	**1 mark** per bullet to **max 3** • Sets up user accounts • Checks usernames, passwords // Authentication • Implements access rights • Automatic backup • System restore / roll back (to previous stable state)	**3**
3(a)(iii)	**1 mark** per bullet to **max 2** • Device / peripheral management • File management • Process management • Input / output management • Error detection / recovery • Provides a user interface • Facilitates communication between hardware and software / hardware devices	**2**

Question 4:

A computer has an operating system (OS) and utility software.

(a) The following table lists key management tasks performed by an operating system and their descriptions.

Complete the table by writing the missing management task names and descriptions.

Management task	Description
Memory management	
	Provides user accounts and passwords
	Handles the signals sent when the attention of the processor is required elsewhere
Provision of a software platform	

[4]

Answer:

Question	Answer	Marks
1(a)	**1 mark** for each correct entry	**4**

Management task	Description
Memory management	Handles the allocation of memory to processes // Ensures two programs do not attempt to use the same memory locations // Keeps track of allocated and free memory locations
Security management	**Provides user accounts and passwords**
Interrupt processing	**Handles the signals sent when the attention of the processor is required elsewhere**
Provision of a software platform	Provides an environment within which programs can be run

Question 5:

(a) Describe the key management tasks of an operating system.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

Answer:

Question	Answer	Marks
4(a)	**1 mark** for identifying task, **max 2** for each description **Max 2** for only identifying tasks without descriptions e.g. • Memory management • Controls the movement of data between RAM, processor, VM etc • allocates memory to processes • File management • Creates files/folders • Renames file/folders • Security management • Creates accounts/passwords • Provide /upgrade firewall / anti-malware • Hardware management • Receives data from input devices ///sends data to output device • Use of device drivers • Process management • Decides which process to run next • supports multitasking	**4**

Question 6:

(a) The operating system (OS) contains code for performing various management tasks.

The appropriate code is run when the user performs various actions.

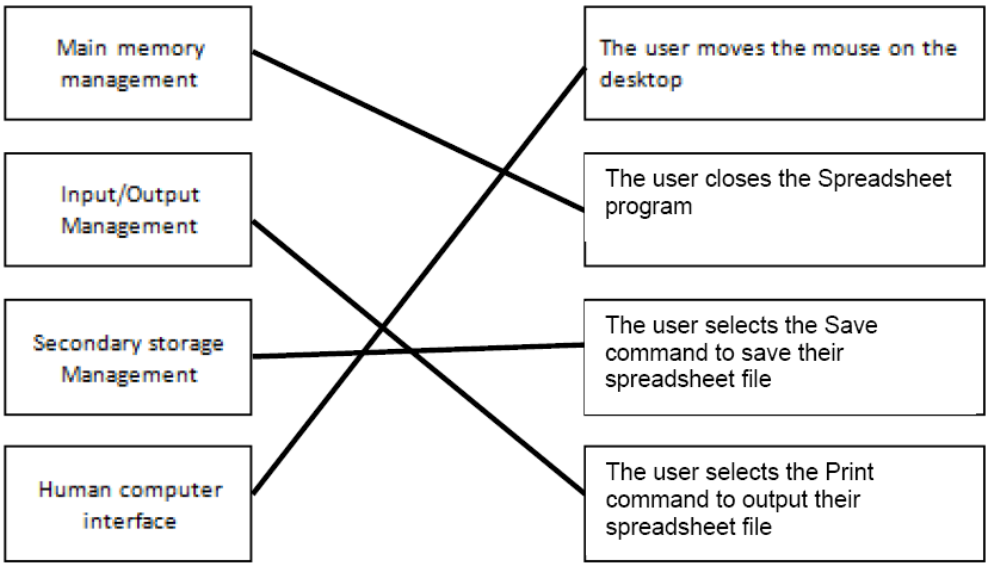
Draw a line to link each OS management task to the appropriate user action.

OS management task	Action
Main memory management	The user moves the mouse on the desktop
Input/Output management	The user closes the spreadsheet program
Secondary storage management	The user selects the Save command to save their spreadsheet file
Human computer interface management	The user selects the Print command to output their spreadsheet document

[3]

Answer:

6(a) — **3**



One mark for each correct line from each left hand box **to max three** marks.

4.1.3 Running of Applications:

NOTE: Running of Applications is a newly added topic in the Computer Science (2210) syllabus for the session 2023–2025.

We need to understand how hardware, firmware and an operating system are required to run applications software.

How hardware, firmware and an operating system are required to run applications software:

The applications are run on the operating system (OS).

- They require the OS to provide a platform on which the software can run successfully.

The operating system (OS) is run on the firmware.

- When a computer starts up, basic input/output system (BIOS) loads bootloader which is referred to as firmware.
- The bootloader loads the part of operating system needed and executes it.
- The BIOS program (bootloader) is stored in a special type of ROM, called an EEPROM (Electrically Erasable Programmable ROM) and its contents remain even when the computer is powered down.

The bootloader (firmware) is run on the hardware

The application software will be under the control of the operating system and will need to access system software such as the device drivers while it is running. Different parts of the operating system may need to be loaded in and out of RAM as the software runs.

The part of the computer of which the bootstrap is an example:

- Firmware

Exam Style Questions:

Question 1:

(d) The computer uses a bootstrap.

Tick (✓) **one** box to show the part of a computer of which the bootstrap is an example.

A application software ☐

B firmware ☐

C hard disk drive ☐

D MAC address ☐

[1]

Answer:

- **B (firmware)**

4.1.4 Interrupts:

- It is a signal sent from a device or from software to the microprocessor that attention is required.
- The processor temporarily stops to service the interrupt.
- The interrupts have different levels of priorities.

The software in the computer that will receive and manage all interrupt signals:

- Operating System OR Interrupt Service Routine (ISR)

Why are Interrupts needed:

- They are needed to stop the current process/task.
- They are needed to allow multitasking.
- They are needed for efficient processing // prioritizing actions.
- They are needed for efficient use of hardware.
- They are needed to allow time-sensitive requests to be dealt with.
- They are needed to avoid the need to poll devices.

What would happen if Interrupt signals were not used in the computer?

- The computer would only start a new task when it had finished processing the current task.
- The computer will not be able to multitask.
- The errors may not be dealt with.
- The computer cannot function without interrupts // computer would become impossible to use.

Examples of when an Interrupt signal could be generated/possible causes of Interrupts:

The interrupts can be software based or hardware based:

Examples of Software Interrupts:

1. Runtime error e.g. division by zero
2. Two processes trying to access the same memory location
3. A running program needs input
4. A buffer is full
5. A buffer requires more data
6. Stack overflow
7. To inform the Central Processing Unit (CPU) that an application has been opened.
8. A phone/video call is received
9. When switching from one application to another

Examples of Hardware Interrupts:

1. Pressing a key on the keyboard
2. Moving the mouse
3. A mouse button is clicked

4. Printer runs out of paper
5. Printer runs out of ink
6. Printer has a paper jam
7. A peripheral is connected/disconnected
8. Power failure
9. No Compact Disc (CD) in drive

Examples of when a printer would generate an Interrupt signal:

1. Paper jam
2. Out of paper
3. Out of ink/toner
4. Buffer full
5. Awaiting input
6. Print complete
7. Printer ready

Examples of devices that make use of Interrupts:

1. Keyboard
2. Printer
3. Mouse

Purpose of an Interrupt in a computer system:

- It is used to attend to certain tasks/issues.
- It is used to make sure that vital tasks are dealt with immediately.
- The interrupt tells the CPU/processor that its attention is required.
- It is a signal that can be sent from a device attached to the computer or from software installed on the computer.
- The interrupt will cause the OS/current process to pause.
- The OS/CPU/ISR will service/handle the interrupt.
- The interrupts have different levels of priority depending upon the task.
- After the interrupt is serviced, the previous process is continued.
- It enables multi-tasking to be carried out on a computer.

For example: It can occur when there is a paper jam in a printer, or 'out of paper' message for a printer or a software error has occurred etc.

When are Interrupts detected in the Fetch-Execute cycle?

- The interrupts are detected at the start/end of a Fetch-Execute cycle.

How the processor handles an Interrupt using Interrupt Service Routine?

- When an interrupt signal is received, the processor checks priority of interrupt.
- If interrupt priority is lower than current process, then it carries on with current task.
- If interrupt priority is higher than current process, then it stops to service the interrupt.
- The processor saves current contents of Program Counter (PC) and other registers.
- It identifies location/source and type of interrupt.
- It calls appropriate Interrupt Service Routine (ISR) to handle the interrupt.
- The address of ISR is loaded into Program Counter (PC).
- When servicing of interrupt is complete it restores saved contents of registers.
- The processor then continues with the next task.

Role of an interrupt in generating a message on the computer that the paper has jammed when a printer begins to print a document:

- The printer generates an interrupt.
- The interrupt is given a priority and queued.
- The interrupt stops CPU from processing current task.
- The Interrupt Service Routine (ISR) services interrupt generating an output message to state there is a paper jam.

The following True & False statements are regarding Interrupts:

Statement	True (✓)	False (✓)
Interrupts can be hardware based or software based	✓	
Interrupts are handled by the operating system	✓	
Interrupts allow a computer to multitask	✓	
Interrupts work out which program to give priority to		✓
Interrupts are vital to computer and it cannot function without them	✓	

The buffers and interrupts allow microprocessor to carry on with multiple tasks (multitasking) thus maximizing its processing power and speed.

Buffers:

- The buffers are used in computers as a small temporary memory area in RAM.
- They are used to store data while it is being moved from one place to another.
- These are essentials in modern computers since hardware device operate at much lower speed than microprocessor.
- The buffers are typically used when there is a difference between the rate at which data is received and the rate at which it can be processed, or in the case that these rates are variable.
- They are used in a printer spooler (as printer speed is much slower than microprocessor speed) or in online video streaming.

Name for the area of memory used to store temporarily the data being sent to the printer:

- Printer buffer.

Exam Style Questions:

Question 1:

Describe the purpose of an interrupt in a computer system.

.....

.....

.....

.....

.....

.....

.....

.....[4]

Answer:

Question	Answer	Marks
8	**Four** from: • Used to attend to certain tasks/issues • Used to make sure that **vital** tasks are dealt with **immediately** • The interrupt/signal tells the CPU/processor (that its attention is required) • A signal that can be sent from a device (attached to the computer) • A signal that can be sent from software (installed on the computer) • The interrupt will cause the OS/current process to pause • The OS/CPU/ISR will service/handle the interrupt • They have different levels of priority • After the interrupt is serviced, the (previous) process is continued • It enables multi-tasking to be carried out on a computer • A valid example of an interrupt e.g. 'out of paper' message for a printer	**4**

Question 2:

Benny is a photographer and prints his photos using an inkjet printer.

(a) Benny is printing some photos and the paper gets jammed in the printer.

A signal is sent to alert the computer about the paper jam.

State the name of this type of signal.

..... [1]

Answer:

- Interrupt

Question 3:

One of the roles of an operating system is to deal with interrupts.

(a) Explain the term interrupt.

.....

.....

.....

..... [2]

(b) Identify **three** devices that make use of interrupts.

Device 1 .....

Device 2 .....

Device 3 .....

[3]

Answer:

6(a)	Any **two** from: – A signal sent from a device / software – Requests processor time // Processor stops to service interrupt – Interrupts have different priorities	2
6(b)	Any **three** from e.g.: – Keyboard – Printer – Mouse	3

Question 4:

(c) An interrupt signal is sent from the printer to the computer.

(i) Give **two** examples of when a printer would generate an interrupt signal.

Example 1 .....

Example 2 .....

[2]

(ii) Many devices send interrupt signals.

Identify the software in the computer that will receive and manage all interrupt signals.

..... [1]

Answer:

Question	Answer	Marks
8(c)(i)	Any **two** from: – Paper jam – Out of paper – Out of toner/ink – Buffer full – Awaiting input – Print complete – Printer ready Award any other valid example	2
8(c)(ii)	Any **one** from: – Operating system – Interrupt handler – Interrupt service routine	1

Question 5:

Padma opens an application on her computer.

An interrupt is generated to inform the Central Processing Unit (CPU) that the application has been opened.

(a) Give **three** other examples of when an interrupt signal could be generated.

1 .....

2 .....

3 .....

[3]

(b) State what would happen if interrupt signals were **not** used in a computer.

.....

..... [1]

Answer:

9(a)	Any **three** from: e.g. – A suitable description of any error that might occur – A peripheral is connected/disconnected – A key on a keyboard is pressed – A mouse button click – A phone/video call is received – A buffer requires more data – A printer has a paper jam – A printer runs out of paper – A printer runs out of ink – When switching from one application to another	3
9(b)	Any **one** from: – The computer would only start a new task when it had finished processing the current task // by example – Computer will not be able to multitask – Errors may not be dealt with – Computer would become impossible to use	1

Question 6:

Frederick prints a document that he has typed.

The printer begins to print the document, but then a message is displayed on Frederick's computer to say that the paper has jammed.

(a) Describe the role of an interrupt in generating a message on the computer that the paper has jammed.

.....

.....

.....

.....

.....

.....

.....

..... [4]

(b) Give **two** other examples of when an interrupt signal could be generated.

1 .....

2 .....
[2]

Answer:

Question	Answer	Marks
4(a)	Any **four** from: – Printer generates interrupt – Interrupt is given a priority – Interrupt is queued – Interrupt stops **CPU** from processing current task – CPU will service interrupt // Interrupt handler services interrupt … – … generating an output message to state there is a paper jam	**4**
4(b)	Any **two** from: – A suitable description of any error that might occur – A peripheral is connected/disconnected – A key on a keyboard is pressed – A mouse button click – A phone/video call is received – A buffer requires more data – A printer runs out of paper – A printer runs out of ink – Opening an application – When switching from one application to another **NOTE:** If two suitable different errors are described, this can be awarded two marks	**2**

Question 7:

A computer has system software.

(a) The Operating System handles interrupts.

Tick (✓) **one** box in each row to identify whether each event is an example of a hardware interrupt or a software interrupt.

Event	Hardware interrupt	Software interrupt
Buffer full		
Printer is out of paper		
User has pressed a key on the keyboard		
Division by zero		
Power failure		
Stack overflow		

[3]

Answer:

Question	Answer	Marks
7(a)	**1 mark** per pair of rows (shaded & unshaded)	**3**

Event	Hardware Interrupt	Software Interrupt	
Buffer full		✓	1
Printer is out of paper	✓		
User has pressed a key on the keyboard	✓		1
Division by zero		✓	
Power failure	✓		1
Stack overflow		✓	

4.2 | Programming Languages, Translators & IDEs

4.2.1 High-Level Language & Low-Level Language:

Computer Program:

- It is a list of instructions that enable a computer to perform a specific task.
- It can be written in:

  1) high-level language
  2) low-level language

High-Level Language:

```
a = input()
b = input()
if a == b:
        print("Correct")
else:
        print("Incorrect")
```

```
FOR X = 1 TO 10
      PRINT X
NEXT X
```

```
FOR x ← 1 TO 10
    READ n
ENDFOR
```

Example of pieces of high-level language codes

Meaning of the term High-Level Language:

- It is closer to human language/English.
- It is machine independent, hence it is a portable language.
- It is problem/logic focused.
- It needs to be translated to low-level language (machine code) for processing by computer.

Examples: C++, Python, Java, Pascal etc.

Advantages of High-Level Language:

1. It is independent of the type of computer being used hence it is portable and works on many different machines/operating systems.
2. It is easier/quicker to read/write/understand codes as uses English-like statements.
3. It is easier and quicker to debug.
4. It is easier to modify.
5. They have built-in functions/libraries which save time when writing the program.
6. It is easier to maintain programs in use.

Disadvantages of High-Level Language:

1. The programmer will not be able to directly manipulate the hardware.
2. The programmer may need to wait for translation before running.
3. The programs may be less efficient and larger.
4. The programs may not be able to make use of special hardware.

Why would a programmer choose to write code in a High-Level Language:

- It is closer to human language/English, so it is easier/quicker to read/write/understand. Therefore, it is less likely to make errors.
- It is easier to debug and maintain the program.
- It is portable as the program can be used on many different platforms because it is written in source code and then it is compiled into object code.
- They have built-in functions/libraries which save time when writing the program.
- There is no need to manipulate memory addresses directly. Therefore, specialist knowledge of manipulating memory locations/registers is not required.
- The programmer only needs to learn a single language as this can be used on many different computers.
- It offers a greater range of programming languages.
- It also offers the use of an Integrated Development Environment (IDE).

Low-Level Language:

Meaning of the term Low-Level Language:

- It is machine code.
- It may use mnemonics.
- It is machine dependent.
- The one line of code in low-level language represents a single instruction.
- It has direct access to memory locations/registers.
- It may need an assembler to be translated.

Examples: assembly code & machine code.

Advantages of Low-Level Language:

1. The programmer will be able to directly manipulate the hardware and make use of special hardware.
2. It can make use of special machine-dependent instructions (machine specific functions).
3. It works directly on registers/CPU and memory locations.
4. It can be executed faster.
5. The translated program requires less memory.
6. The programmer has more control over what happens on the computer.

Disadvantages of Low-Level Language:

1. It is more difficult to read/write/understand codes.
2. It is error prone and difficult to debug codes.
3. It takes a longer time to write programs.
4. It is machine dependent and so not portable.
5. The programmer will have to manipulate memory locations.

Why would a programmer choose to write code in a Low-Level Language:

- It allows direct access to a computer processor and allows the use of special hardware and machine-dependent instructions.
- The program code written in low-level language uses up less memory.
- The program code written in low-level language allows faster execution of instructions.

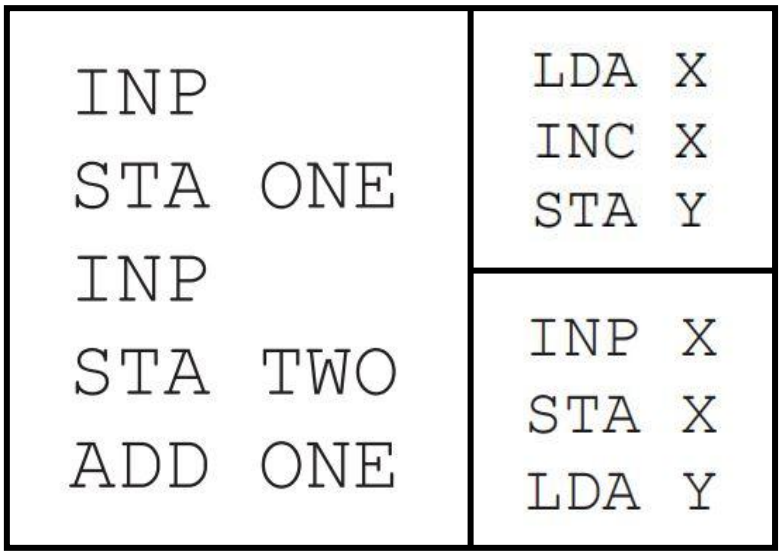
Machine Code:

1 0 1 0 1 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1	10110111 11001100 01011100	1011100000110000 0000011011100010

Example of pieces of machine code

- It is the binary language/instructions that the computer uses and understands.
- The machine code requires no translation to be understood by the computer.

4.2.2 Assembly Language:



```
INP
STA ONE
INP
STA TWO
ADD ONE
```

```
LDA X
INC X
STA Y
```

```
INP X
STA X
LDA Y
```

Example of pieces of assembly language codes

- It is a form of low-level programming language.
- It uses mnemonic codes.
- An assembler is needed to translate an assembly language program into machine code.
- It is specific to the computer hardware.
- It is not machine code at all and instead it needs to be converted into machine code for the computer to understand.
- It is often used to create drivers for hardware.

The following True & False statements are regarding Assembly Language:

Statement	True (✓)	False (✓)
Assembly language uses mnemonic code	✓	
Assembly language programs do not need translator to be executed		✓
Assembly language is a low-level programming language	✓	
Assembly language is specific to the computer hardware	✓	
Assembly language is machine code		✓
Assembly language is often used to create drivers for hardware	✓	

The following True & False statements are regarding High-Level Languages:

Statement	True (✓)	False (✓)
High-level languages need to be translated into machine code to run on a computer	✓	
High-level languages are written using mnemonic codes		✓
High-level languages are specific to the computer's hardware		✓
High-level languages are portable languages	✓	

The following table compares High-Level Language, Assembly Language & Machine Code:

Statement	High-Level Language (✓)	Assembly Language (✓)	Machine Code (✓)
It requires a translator to be processed by a computer	✓	✓	
It is an example of low-level language		✓	✓
It uses mnemonics		✓	
It uses English-like statements	✓		
It is portable	✓		
It can be used to directly manipulate hardware in the computer		✓	✓

Exam Style Questions:

Question 1:

High-level or low-level languages can be used when writing a computer program.

State **two** advantages of using a high-level language and **two** advantages of using a low-level language.

High-level language advantage 1 .....

.....

High-level language advantage 2 .....

.....

Low-level language advantage 1 .....

.....

Low-level language advantage 2 .....

.....

[4]

Answer:

Any **two** from:

High level language

- easier/faster to write code as uses English-like statements
- easier to modify as uses English-like statements
- easier to debug as uses English-like statements
- portable language code

Any **two** from:

Low level language

- can work directly on memory locations
- can be executed faster
- translated program requires less memory [4]

Question 2:

A programmer uses a high-level language to create a computer program.

(a) (i) Identify **two** advantages to the programmer of using a high-level language instead of a low-level language.

1 .....

2 ..... [2]

(ii) Suggest **one** disadvantage to the programmer of using a high-level language instead of a low-level language.

..... [1]

Answer:

5(a)(i)	Any **two** from: • it is easier / quicker to read/write/understand • it is easier / quicker to debug • code is portable.	**2**
5(a)(ii)	Any **one** from: • not able to directly manipulate the hardware • may need to wait for translation before running • program may be less efficient.	**1**

Question 3:

(a) Identify **three** features of a low-level language.

Feature 1 .....

Feature 2 .....

Feature 3 .....
[3]

(b) Give **two** examples of a low-level language.

Example 1 .....

Example 2 .....
[2]

(c) Give **one** drawback of writing programs in a low-level language, instead of a high-level language.

.....

..... [1]

Answer:

Question	Answer	Marks
9(a)	Any **three** from: – Closer to/is machine code – May use mnemonics – May need an assembler to be translated – One line of code represents a single instruction – Machine dependent – Have direct access to memory locations/registers	3
9(b)	– Assembly code – Machine code	2
9(c)	Any **one** from: – It is more difficult to understand – Error prone – Have to manipulate memory locations – Machine dependent	1

Question 4:

(e) Study the following three sections of code.

A:
```
1 0 1 0 1 1 0 1
1 1 0 0 1 1 1 0
1 0 1 1 0 1 1 1
```

B:
```
LDA X
INC X
STA Y
```

C:
```
FOR x ← 1 TO 10
   READ n
ENDFOR
```

Identify, using the letters A, B or C, which of the above codes is an example of assembly code, high-level language code or machine code:

Assembly code .....

High-level language code .....

Machine code .....

[2]

Answer:

(e) 1 mark per correct letter, maximum 2 marks

Assembly code: **B**
High-level language code: **C**
Machine code: **A**

Question 5:

Annie writes a paragraph of text as an answer to an examination question about programming languages.

Using the list given, complete Annie's answer by inserting the correct **six** missing terms. Not all terms will be used.

- Assembly
- Converter
- Denary
- Hexadecimal
- High-level language
- Low-level language
- Machine Code
- Source Code
- Syntax
- Translator

The structure of language statements in a computer program is called the

..... . A programming language that uses natural

language statements is called a ..... . When programs

are written in this type of language they need a ..... to

convert them into ..... .

A programming language that is written using mnemonic codes is called a

..... . An example of this type of language is

..... language.

[6]

Answer:

Question	Answer	Marks
7	1 mark for each correct term, in the correct place: • Syntax • High-level language • Translator • Machine code • Assembly • Low-level language	6

Question 6:

(c) **Three** examples of computer code are given in the table.

Tick (✓) to show whether each example of computer code is **High-level language**, **Assembly language** or **Machine code**.

Computer code	High-level language (✓)	Assembly language (✓)	Machine code (✓)
10110111 11001100 01011100			
FOR X = 1 TO 10 PRINT X NEXT X			
INP X STA X LDA Y			

[3]

Answer:

Question	Answer	Marks
6(c)	1 mark for each correct tick (✓)	3

Computer code	High-level language (✓)	Assembly language (✓)	Machine code (✓)
10110111 11001100 01011100			✓
FOR X = 1 TO 10 PRINT X NEXT X	✓		
INP X STA X LDA Y		✓	

Question 7:

(b) Tick (✓) to show which of the following is an example of a high-level language program.

Example program	Tick (✓)
1011100000110000 0000011011100010	
INP STA ONE INP STA TWO ADD ONE	

a = input()
b = input()
if a == b:
  print("Correct")
else:
  print("Incorrect")	

[1]

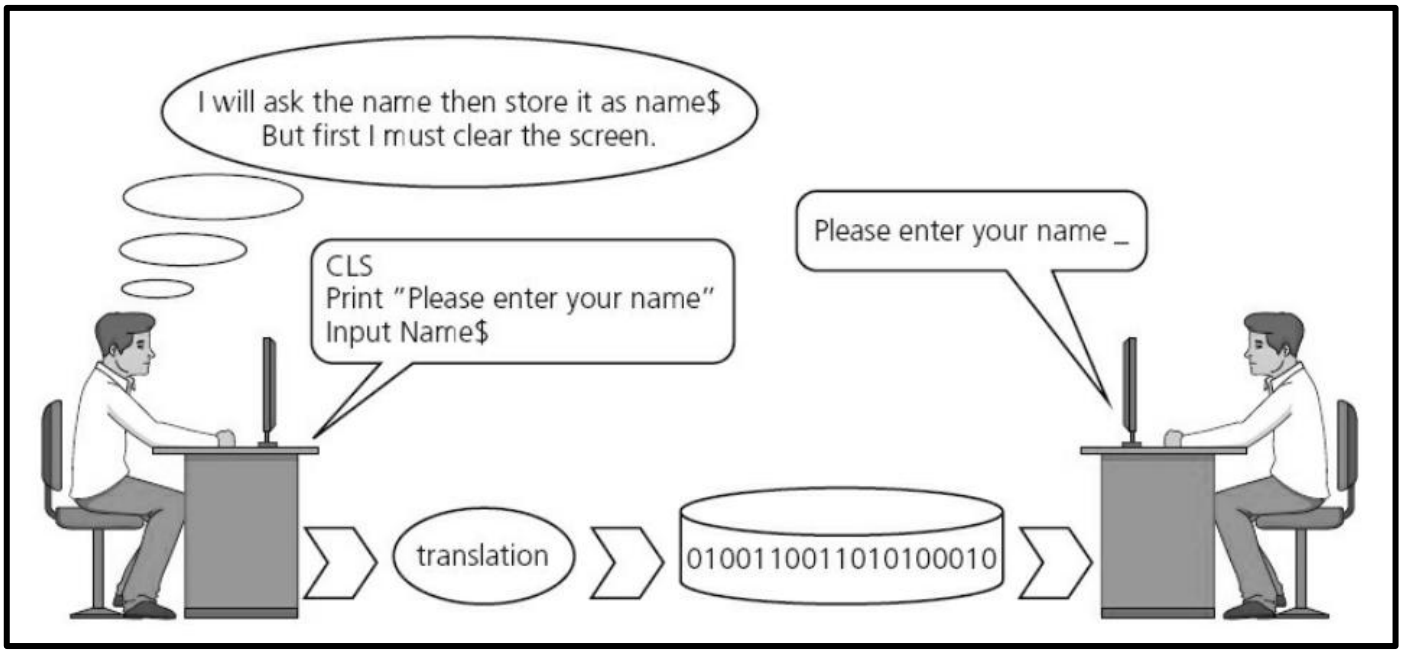
Answer:

Question	Answer	Marks
2(b)	**One** mark for the correct tick	1

Example program	Tick (✓)
1011100000110000 0000011011100010	
INP STA ONE INP STA TWO ADD ONE	

a = input()
b = input()
if a == b:
  print("Correct")
else:
  print("Incorrect")
	✓

4.2.3 Translators:



It is a utility program that converts a program written in high-level or low-level language into binary instructions that can be understood/used by the computer.

Types of Translators:

Compiler:

```
Sum := FirstNumber + SecondNumber
```

becomes the following machine code instructions when translated

0001	00010010
0100	00010011
0000	00011010

- It translates high-level language into low level language.
- It translates the entire source code into machine code/object code all at once/in one go.
- It produces an executable file in machine code that runs directly on the computer.
- It is more difficult to modify the executable code since it is in machine code format.
- It generates an error report at the end of translation of the whole program.
- Once translated, the translator does not need to be present for the program to run.
- It is more likely to crash the computer since the machine code produced runs directly on the processor.
- A compiled program is usually distributed for general use.

Working/Operation of Compiler:

- It translates high-level language into machine code/low level language.
- It translates the whole program (the source code) as a complete unit/all at once/all in one go.
- It creates an executable file/object code.
- It provides an error report/list of errors for the whole code if errors are detected.
- It optimizes the source code to run efficiently.

Interpreter:

- It translates high-level language into low level language.
- It executes a high-level language program one instruction/statement at a time.
- It translates source code line by line.
- It stops the execution as soon as the first error is encountered.
- It continues translating when an error is fixed.
- It allows slow speed of execution of program loops.
- It is easier to debug since each line of code is analyzed and checked before being executed.
- An interpreter is often used when developing a program.

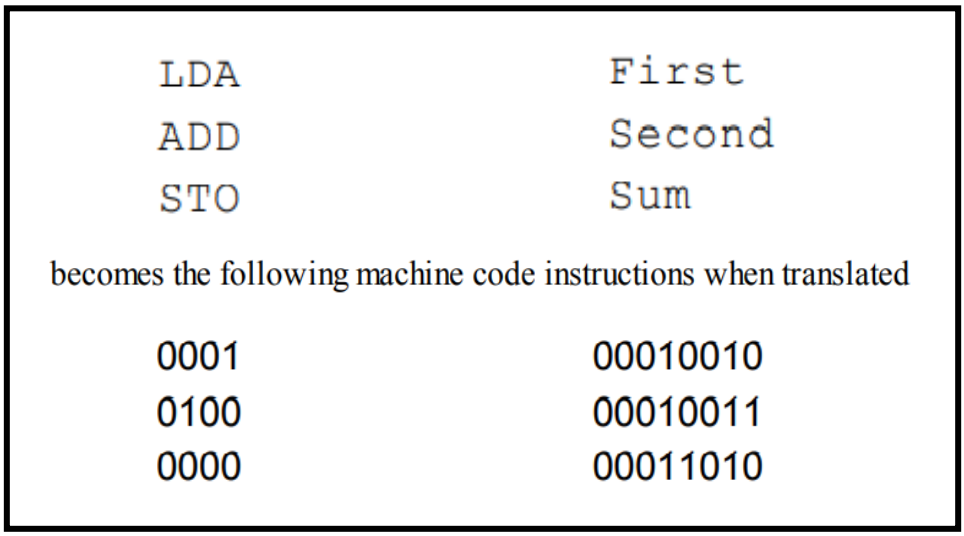
Working/Operation of Interpreter:

- It translates high-level language into machine code/low level language.
- It translates a program (the source code) one line of code at a time/line by line/statement by statement.
- It stops execution when it encounters an error and will only continue translating when the error is fixed.
- It produces error messages to tell the user the exact location of the error.
- The machine code is directly executed which means the interpreter is used each time the program/code is executed.

Why a compiler or an interpreter is needed when running a high-level program on a computer:

- The computer only understands machine code/binary language.
- The compiler/interpreter converts the code into machine code/binary language.

Assembler:



- It translates a program written in low-level language (assembly language) into machine code.
- An executable file is produced in machine code.
- It translates the entire source code into machine code/object code all at once/in one go.
- It produces error messages and provides error diagnostics.
- The assembled programs can be used without the assembler.
- An assembled program is usually distributed for general use.

Summary of Translators:

Compiler	Interpreter	Assembler
Translates a high-level language program into machine code.	Executes a high-level language program a statement at a time.	Translates a low-level language program into machine code.
An executable file of machine code is produced.	No executable file of machine code is produced.	An executable file of machine code is produced.
One high-level language statement can be translated into several machine code instructions.	One high-level language program statement may require several machine code instructions to be executed.	One low-level language statement is usually translated into one machine code instruction.
Compiled programs are used without the compiler.	Interpreted programs cannot be used without the interpreter.	Assembled programs can be used without the assembler.
A compiled program is usually distributed for general use.	An interpreter is often used when a program is being developed.	An assembled program is usually distributed for general use.

Compiler & Interpreter:

An interpreter is mostly used when developing a program and a compiler is used to translate the final program.

Why a programmer would make use of both an interpreter and a compiler when writing a program in a high-level language:

- Both are used to translate the high-level language into machine code for the computer to understand.
- The interpreter is used whilst writing the program and to debug the code line by line.
- The compiler is used when the program is completed and to create a separate executable file (so it no longer needed for the execution of program).
- If the program runs for the first time in a compiler, then it means there are no syntax errors.

Main Similarities between a Compiler & an Interpreter:

1. They both translate high-level language/source code into machine code/low-level language.
2. They both check for errors.
3. They both identify and report errors generating error diagnostics/messages.

Main Differences between a Compiler & an Interpreter:

1. An interpreter translates and executes the code line by line whereas a compiler translates and executes the whole code all in one go.
2. An interpreter stops translating and reports an error as it finds one whereas a compiler produces an error report/list of errors at the end of translation.
3. An interpreter does not produce an executable file, but a compiler does produce an executable file.
4. An interpreter will execute the code until it finds an error whereas a compiler will not execute any code if there are error present.
5. An interpreter allows correction of errors in real-time whereas compiler needs to retranslate the code each time after errors are found and corrected.

4.2.4 Advantages & Disadvantages of Compiler & Interpreter:

An interpreter is mostly used when developing a program and a compiler is used to translate the final program.

Advantages of Compiler:

1. The compiler produces an executable file which creates a smaller file size.
2. The user does not have access to source code so users are unable to make changes to the program/modify it.
3. The compiled program will be machine independent/portable which means it can be used on any hardware.
4. The compiled code executes faster than the equivalent for an interpreter.
5. The code does not have to be recompiled each time it is run.
6. Once translated, the compiler is not required to run the program.
7. All errors are reported in a single report meaning they can all be fixed at the same time.
8. The program can be distributed without the source code and so source code is kept secure (cannot be stolen or plagiarized).
9. The cross-compilation is possible.

Disadvantages of Compiler:

1. The program will not run if there are any errors.
2. The program must be recompiled after every change as errors cannot be corrected in real-time.
3. The parts of the program cannot be tested, without all the program code being available.
4. One error may result in other false errors being reported.

Advantages of Interpreter:

1. It allows easier debugging as errors are immediately reported when detected and they can be corrected in real time as they occur.
2. The errors are reported as the interpreter finds them.
3. An error can be corrected, and translation continued from where it stopped.
4. The effect of any change made to the code can be seen immediately.
5. It can run a partially complete program during development stage.
6. The parts of the program can be tested, without all the program code being available.

Disadvantages of Interpreter:

1. The source code is needed at run time.
2. No executable file is produced so source code can be edited and modified.
3. The translation software (interpreter) is needed every time the program is run.
4. The programs can take longer to execute than the equivalent for a compiler.

A few examples are given below which will help you better understand the choice of each translator according to the required use (scenario) and the benefits being provided by that translator for choosing that particular translator according to the examination question.

Example 1:

A program is written using a high-level language so it will be published and sold for profit. The program is compiled when it is completed.

Benefits of compiling the program:

1. It produces an executable file which creates a smaller file size that is more saleable.
2. The program will be machine independent/portable which means it can be used on any hardware.
3. There is no need for a compiler to run executable file which means it will be quicker to run and the customers can just execute the program.
4. The source code cannot be accessed and therefore code cannot be stolen or plagiarized.

Example 2:

Which translator is the most suitable for the given tasks along with benefits of using that translator for the task:

(i) To translate the code during development of the game:

Interpreter:

- It is easier to debug as errors are immediately reported when detected.

Compiler:

- All errors are reported in a single report meaning they can all be fixed at the same time.
- There is no need to recompile code every time a test is run.

(ii) To translate the final program and upload to the website for distribution, without the source code:

Compiler:

- It creates an executable file, so a translator is no longer needed to run it.
- The source code cannot be stolen as it can be provided without the source code.

A few examples are given below which will help you better understand the choice of each translator according to the required use (scenario) and the explanation needed for choosing that particular translator according to the examination question.

Example 1:

Dimitri is writing a computer program in a high-level language. He needs to send just the machine code for the program to his friend, electronically. It is important that the program is executed as quickly as possible.

Identify which translator will be most suitable for Dimitri to use. Explain your choice.

Compiler:

- It does not require recompilation as compiled program can be executed without a compiler. Therefore, it allows faster execution.
- It provides an executable file hence allows him to just send machine code.
- Dimitri's friend does not need translation/compilation software to execute the program.

Example 2:

Julius uploads his application to his website for people to download. Before he uploads the application, he translates the code using a compiler.

Why Julius uses a compiler, rather than an interpreter, to do this:

- It creates an executable file so he would not release source code.
- The source code therefore cannot be stolen/edited.
- The program would not need to be translated every time, so a translator is not required.
- This makes it machine independent.

Example 3:

Three programmers are working on different projects:

- **Alice is developing a program written in a low-level language**
- **Akbar is developing a program written in a high-level language**
- **Alex is preparing a program written in a high-level language for sale**

State, with reasons, which type of translator each programmer should use. Each programmer should be using a different type of translator:

Alice:

- Assembler because it translates low-level language into machine code.

Akbar:

- Interpreter because it is easier to identify where an error is and easier to debug a program.

Alex:

- Compiler because once translated, a stand-alone program file is created and there is no need for the compiler when running the program so it can be distributed for general use and sales.

Example 4:

Ishani uses an interpreter and compiler at different stages of game creation.

(i) When it is most appropriate for Ishani to use an interpreter:

- It is most appropriate during development // when writing the program // when debugging.
- The interpreter makes the code easier to debug.
- It stops when an error is detected and reports one error at a time.
- It can correct errors in run-time.
- It can test one section without the rest of the code being completed.

(ii) When it is most appropriate for Ishani to use a compiler:

- It is most appropriate after completion // for distribution // for final testing.

After completion, the compiler creates an executable file that can be distributed without source code:

- The source code cannot be edited/viewed by other people.
- End users do not need translator software and so the game is machine/platform independent.

In final testing, the compiler creates an executable file and so code does not need to retranslate for each test sequence therefore it can be tested repeatedly with different data faster.

The following tables compare Interpreter & Compiler:

(i) Table 1:

Statement	Interpreter (✓)	Compiler (✓)
takes one statement at a time and executes it	✓	
generates an error report at the end of translation of the whole program		✓
stops the translation process as soon as the first error is encountered	✓	
slow speed of execution of program loops	✓	
translates the entire program in one go		✓

(ii) Table 2:

Statement	Interpreter (✓)	Compiler (✓)
creates an executable file that runs directly on the computer		✓
more likely to crash the computer since the machine code produced runs directly on the processor		✓
easier to debug since each line of code is analyzed and checked before being executed	✓	
slow speed of execution of program loops	✓	
it is more difficult to modify the code since the executable code is now in the machine code format		✓

(iii) Table 3:

Statement	Interpreter (✓)	Compiler (✓)
Translates the source code into machine code all at once		✓
Produces an executable file in machine code		✓
Executes a high-level language program one instruction at a time	✓	
Once translated, the translator does not need to be present for the program to run		✓
An executable file is produced		✓

(iv) Table 4:

Statement	Interpreter (✓)	Compiler (✓)
It is more difficult to debug the code since one error can produce many other associated errors		✓
The speed of execution of program loops is slower	✓	
It produces fast, executable code that runs directly on the processor		✓
It is easier to debug the code since an error is displayed as soon as it is found	✓	

The following tables compare Assembler, Compiler & Interpreter:

(i) Table 1:

Statement	Assembler (✓)	Compiler (✓)	Interpreter (✓)
Translates high-level language into machine code		✓	✓
Provides error diagnostics	✓	✓	✓
Translates whole program to object code in one operation	✓	✓	
Translates and executes one line of code at a time			✓

(ii) Table 2:

Statement	Assembler (✓)	Compiler (✓)	Interpreter (✓)
Translates low-level language to machine code	✓		
Translates high-level language to machine code		✓	✓
Produces error messages	✓	✓	✓
Translates high-level language one line at a time			✓
Produces an executable file	✓	✓	

Exam Style Questions:

Question 1:

Describe how a compiler and an interpreter translates a computer program.

Compiler .....

.....

.....

.....

.....

.....

Interpreter .....

.....

.....

.....

.....

.....

[6]

Answer:

Question	Answer	Marks
7	Compiler Any **three** from: – Translates high-level language into **machine code/low level language** – Translates (the source code) all in one go/all at once – Produces an executable file – Produces an error report Interpreter Any **three** from: – Translates high-level language into **machine code/low level language** – Translates (the source code) line by line/statement by statement – Stops if it finds an error – Will only continue when error is fixed	**6**

Question 2:

Both an interpreter and a compiler can be used when writing a program in a high-level language.

(a) Explain why a programmer would make use of both an interpreter **and** a compiler.

.....

.....

.....

.....

.....

.....

.....

..... [4]

Answer:

Question	Answer	Marks
2(a)	Any **four** from: – To translate the high-level language into low-level language – Interpreter used whilst writing the program – Interpreter used to debug code line by line – Compiler used when program completed – Compiler used to create separate executable file (so compiler no longer needed) – If it runs first time in a compiler there are no syntax errors	4

Question 3:

(a) Give **one** similarity between a compiler and an interpreter.

.....

..... [1]

(b) Describe **two** differences between a compiler and an interpreter.

Difference 1 .....

.....

.....

.....

Difference 2 .....

.....

.....

.....

[4]

(c) Identify **one** other type of translator.

..... [1]

Answer:

Question	Answer	Marks
6(a)	Any **one** from: – They both translate **high-level language** into **machine code** / **low-level language** – They both check for errors – They both report errors	1
6(b)	**Four** from (Max 2 per translator): – An interpreter translates and executes the code line by line – … whereas a compiler translates and executes the whole code all in one go – An interpreter stops translating and reports an error as it finds one – … whereas a compiler produces an error report at the end of translation – An interpreter does not produce an executable file – … but a compiler does produce an executable file – An interpreter will execute the code until it finds an error – … whereas a compiler will not execute any code if there are errors present – An interpreter allows correction of errors in real-time – … whereas a compiler needs to retranslate the code each time after errors are found and corrected	4
6(c)	– Assembler	1

Question 4:

(b) Adeel uses an interpreter while developing and testing the application.

Adeel uses a compiler when the application is ready to be shared with others.

Compare the features of interpreters and compilers.

.....

.....

.....

.....

.....

.....

.....

..... [4]

Answer:

7(b)	**Four** from **Max 2** for only giving compiler/interpreter features – **Both** translate high level / source code to machine code – **Both** generate error diagnostics/messages // identify errors – Interpreter translates one line at a time // checks one line and then runs it – Compiler translates whole code in one go // checks all code and then runs it – Interpreter **stops** when meets an error – …and then allows you to continue running from where you stopped // correct errors in real-time – Compiler provides list of all errors – Interpreter does not produce an **executable** file – Compiler produces an **executable** file	**4**

Question 5:

David is writing a program using a high-level language. The program will be published and sold for profit.

(b) David compiles the program when he has completed it.

Explain **two** benefits of compiling the program.

Benefit 1 .....

.....

.....

.....

Benefit 2 .....

.....

.....

.....

[4]

Answer:

7(b)	**Four** from (Max three per benefit): - Produces executable file … - … this creates a smaller file size - … more saleable - Program will be machine independent / portable … - … this means it can be used on any hardware - No need for compiler to run executable file … - … this means it will be quicker to run - … customers can just execute the program - Source code cannot be accessed … - … therefore, code cannot be stolen / plagiarised	**4**

Question 6:

Ishan is a member of a software community that develops computer games. He has programmed a new feature for one of the community's existing games.

(a) Ishan compiles the program before he issues it to the community.

(i) Explain **one** benefit of Ishan compiling the program.

.....

..... [1]

(ii) Explain **one** drawback of Ishan compiling the program.

.....

..... [1]

Answer:

Question	Answer	Marks
6(a)(i)	**One** from: • Code will run without the need of an interpreter • (Object) Code is platform independent • Source code not available / cannot be modified	1
6(a)(ii)	**One** from: • Source code not available / cannot be modified • Comments, etc. not visible • Future changes will require code to be recompiled	1

Question 7:

(d) Francis's team use language translators.

Complete the descriptions of language translators by writing the missing words.

..... are usually used when a high-level language program is complete. They translate all the code at the same time and then run the program.

They produce ..... files that can be run without the source code.

..... translate one line of a high-level language program at a time, and then run that line of code. They are most useful while developing the programs because errors can be corrected and then the program continues from that line.

Assemblers are used to translate assembly code into ..... .

[4]

Answer:

4(d)	**1 mark** for each correctly completed term **Compilers** are usually used when a high-level language program is complete. They translate all the code at the same time and then run the program. They produce **executable/.exe/object code** files that can be run without the source code. **Interpreters** translate one line of a high-level language program at a time, and then run that line of code. They are most useful while developing the programs because errors can be corrected and then the program continues from that line. Assemblers are used to translate assembly code into **binary/machine code**.	**4**

Question 8:

(b) A programmer is developing software and has both a compiler and interpreter for the high-level language used.

Describe **two** benefits of using each form of translation software.

(i) Benefits of a compiler

1 .....

.....

2 .....

.....

[2]

(ii) Benefits of an interpreter

1 .....

.....

2 .....

.....

[2]

Answer:

2(b)(i)	**1 Mark** per bullet, max 2 • Once translated the compiler software is not needed to run the program • Compiled code should execute faster • Compiler produces an executable file • The executable file produced by a compiler can be distributed without users having sight of the source code // source code is kept secure // users are unable to make changes to the program • Cross-compilation is possible	**2**
2(b)(ii)	**1 Mark** per bullet, max 2 • Easier de-bugging • The interpreter stops when error encountered • error can be corrected in real time • The interpreter translates a statement then executes it immediately • Parts of the program can be tested, without all the program code being available.	**2**

Question 9:

Kimmy has written a program in a high-level language.

(b) Three translators are compilers, interpreters, and assemblers.

(i) State **one** benefit of Kimmy using an **interpreter** during the development of the program.

.....

..... [1]

(ii) State **three** benefits of Kimmy using a **compiler** when the program is complete.

1 .....

.....

2 .....

.....

3 .....

.....

Answer:

3(b)(i)	**1 mark** per bullet point to **max 1** □ Errors can be corrected as they occur □ Can run a partially complete program when developing □ The effect of any change made to the code can be seen immediately	**1**
3(b)(ii)	**1 mark** per bullet point to **max 3** □ Produces an executable file □ User does not have access to source code □ It will (probably) be faster to run the executable □ Code does not have to be compiled each time it is run □ Does not need the compiler to be present at run-time	**3**

Question 10:

Patrick is writing a new software application. He is using a compiler to develop the software application.

(a) Describe the drawbacks of using a compiler instead of an interpreter.

.....

.....

.....

.....

.....

..... [3]

Answer:

Question	Answer	Marks
7(a)	**1 mark** per bullet point to **max 3** • The program will not run if there are any errors • The program must be recompiled after every change // cannot correct errors in real-time • Part-programs cannot be tested • One error may result in other false errors being reported	**3**

Question 11:

(a) State **three** benefits of using an interpreter.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(b) State **one** drawback of using an interpreter.

.....

..... [1]

Answer:

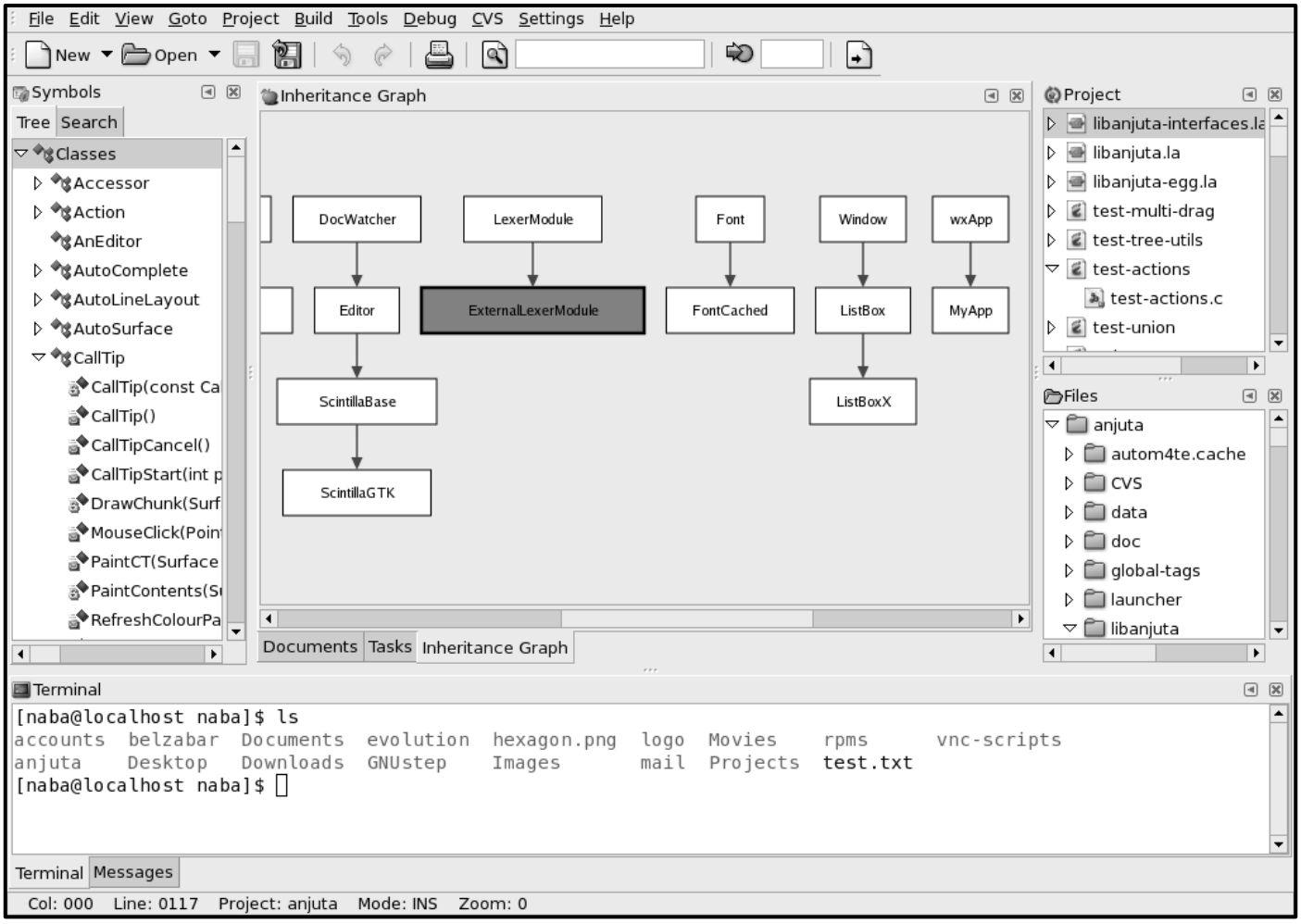
Question	Answer	Marks
10(a)	**One mark** per bullet point to **max 3** • Easier de-bugging • Errors can be corrected in real time • Errors are reported as the interpreter finds them. • An error can be corrected and translation continued from where it stopped • The effect of any change made to the code can be seen immediately • Parts of the program can be tested, without all the program code being available.	**3**

10(b)	**One mark** for a correct answer
	
	Source code is needed at run time
	// No executable file produced, (so source code can be edited)
	// Translation software needed every time the program is run
	// execution time increased
		1

4.2.5 Integrated Development Environment (IDE):

NOTE: Integrated Development Environment (IDE) is a newly added topic in the Computer Science (2210) syllabus for the session 2023–2025.

- It is a software that provides useful functions for a programmer writing a computer program.
- It helps the programmers in writing and development of programs.
- Most high-level programming languages offer the use of an IDE for program development.
- This contains source code editor with an interpreter and/or compiler together with debugging tools, which can improve the speed of program development.



Example of an IDE

Functions/Features of IDEs:

1. Code editors
2. Translators
3. A runtime environment with a debugger
4. Error diagnostics
5. Auto-completion
6. Auto-correction
7. Prettyprint

Description of IDEs Functions:

1) Code editors:

- A code editor allows a program to be written and edited without the need to use a separate text editor.
- This speeds up the program development process, as editing can be done without changing to a different piece of software each time the program needs correcting or adding to.

2) Translators:

- Most IDEs usually provide a translator, this can be a compiler and/or an interpreter, to enable the program to be executed.
- An interpreter is mostly used when developing a program.
- A compiler is used to translate the final program to be used.

3) A runtime environment with a debugger:

- A debugger is a program that runs the program under development.
- It allows the programmer to step through the program a line at a time (single stepping) or to set a breakpoint to stop the execution of the program at a certain point in the source code.
- A report window then shows the contents of the variables and expressions evaluated at that point in the program.
- This allows the programmer to see if there are any logic errors in the program and check that the program works as intended.

4) Error diagnostics & auto-correction:

- The dynamic error checking finds possible errors as the program code is being typed, alerts the programmer at the time, and provides a suggested correction.
- Many errors can therefore be found and corrected during program writing and editing before the program is run.

5) Auto-completion:

- The code editors can offer context-sensitive prompts with text completion for variable names and reserved words.

6) Prettyprint:

- This is color-coding of keywords, built-in function calls, comments, strings, and the identifier in a function header.
- It presents characters with line breaks and indentations to make code comprehensible.
- It makes source code easier to read due to better presentation.
- It also increases the ability to reuse strings.

How a programmer can make use of an Integrated Development Environment (IDE) when writing and testing a program:

(i) Writing a program:

- The programmer can enter code into an editor.
- The programmer can use pretty printing to identify key terms.
- The programmer can use context-sensitive prompts to help complete statements.
- The programmer can expand and collapse code blocks.
- The programmer can use auto-complete to suggest what to type next.
- The programmer can use auto-formatting to indent code blocks.
- The programmer can use dynamic syntax checking.

(ii) Testing a program:

- The programmer can use single stepping to run the code line by line.
- The programmer can set breakpoints to stop the code at set points to check values.
- The programmer can use report window to see how variables change.

Exam Style Questions:

Question 1:

(b) The programmer uses an integrated development environment (IDE) when creating the computer program.

State what is meant by an IDE.

.....

..... [1]

Answer:

- **Software that provides useful functions for a programmer writing a computer program.**

Question 2:

A programmer uses an Integrated Development Environment (IDE) to develop a program.

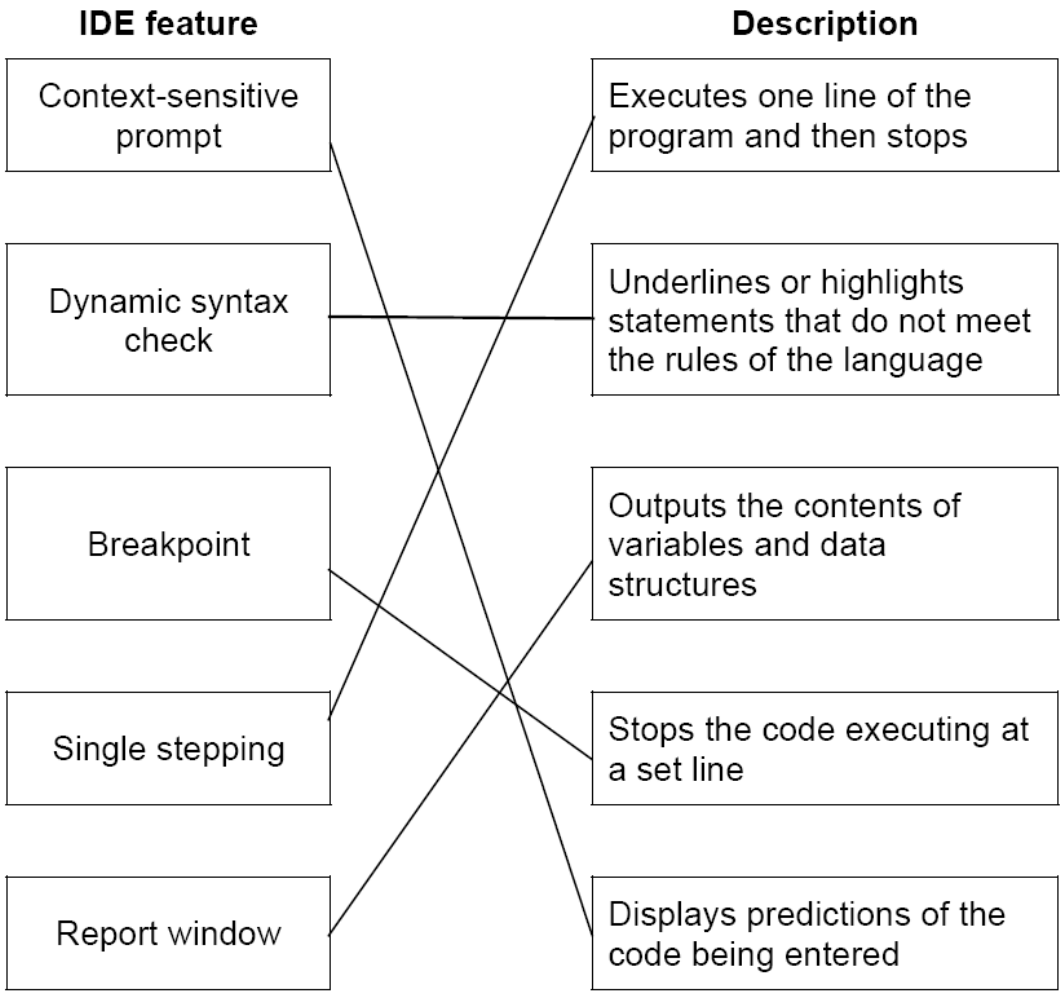
(a) Draw **one** line from each IDE feature to its correct description.

IDE feature	Description
Context-sensitive prompt	Executes one line of the program and then stops
Dynamic syntax check	Underlines or highlights statements that do not meet the rules of the language
Breakpoint	Outputs the contents of variables and data structures
Single stepping	Stops the code executing at a set line
Report window	Displays predictions of the code being entered

[4]

Answer:

Question	Answer	Marks
5(a)	**1 mark** for 1 correct line, **2 marks** for 2 correct lines, **3 marks** for 3 or 4 correct lines, **4 marks** for all 5 correct lines	**4**



Question 3:

(ii) A typical IDE provides debugging tools to support the testing of a program.

Identify **three** other tools or features found in a typical IDE to support the writing of the program.

1 .....

2 .....

3 .....

[3]

Answer:

4(b)(ii)	**1 mark** for each correct tool e.g. • Colour coding // pretty printing • Auto-complete • Auto-correct • Context sensitive prompts • Expand and collapse code blocks	**3**