OCR A Level Computer Science

Your notes

2.2 Applications Generation

Contents

Application Software

Your notes

Application Software

What are Applications?

- Applications are software designed to perform a specific task or tasks for a user
- Application types include word processors, database management, web browsers, graphics manipulation, games and many more
- Application software enables users to perform tasks like creating documents, managing data, and surfing the web

Common types of application software

Type	Purpose	Examples
Word Processor	For creating, editing, formatting and printing text documents.	Microsoft Word, Google Docs
Database Management Software	For storing, retrieving, manipulating and managing data.	MySQL, Oracle
Web Browser	For browsing the web and accessing information online.	Google Chrome, Firefox
Graphics Manipulation	For editing and manipulating images, photos and graphic designs.	Adobe Photoshop, GIMP
Spreadsheet Software	For organising, calculating and analysing numerical data.	Microsoft Excel, Google Sheets
Presentation Software	For creating visual and multimedia presentations.	Microsoft PowerPoint, Keynote
Antivirus Software	To protect the computer system from viruses and other malicious software.	Norton, McAfee

Email Client	For managing and accessing email across different email providers.	Outlook, Thunderbird
Video Editing Software	For editing and producing videos and films.	Adobe Premiere, Final Cut Pro
Integrated Development Environment (IDE)	For developing, testing and debugging software code.	Visual Studio, IntelliJ IDEA
Virtualisation Software	For creating and running virtual machines on a single physical machine.	VMware, VirtualBox

Your notes

Examiner Tips and Tricks

Exam questions on this topic typically present a scenario and then require a recommendation of suitable software.

How to recommend an application:

1. Carefully read the scenario and identify the specific tasks the user must perform
2. Identify two or more software applications that have features to complete these tasks
3. Write your recommendation and clearly state how the application feature helps achieve the user needs from the scenario

Worked Example

'Bee Bank' is a new online bank launching next year. The directors of the bank have set a target to open 10,000 new accounts in the first month. To open accounts for new customers, the bank needs to write to them by letter and store their data somewhere.

Recommend two types of application software for the bank and explain your choice.

[4]

How to answer this question:

- Identify user tasks from the scenario
- Recall software applications that would assist in completing these tasks

- Clearly state what feature about the application helps the user achieve these tasks

Answer:

Example answer that gets full marks:

For Bee Bank, I recommend the following two types of application software:

- Word Processor: A word processor like Microsoft Word is ideal for creating, editing, formatting, and printing letters, meeting the bank's need to communicate with new customers.
- Database Management System (DBMS): A DBMS like Oracle Database offers robust data management features, including secure data storage, retrieval, and manipulation, meeting the bank's need to hold and manage customer data.

Acceptable answers you could have given instead:

For the new bank, I recommend using a Word Processor such as Microsoft Word to create letters to send to customers and a Database Management System (DBMS) like Oracle Database to securely store and manage customer data.

Your notes

Utility Software

Your notes

Utility Software

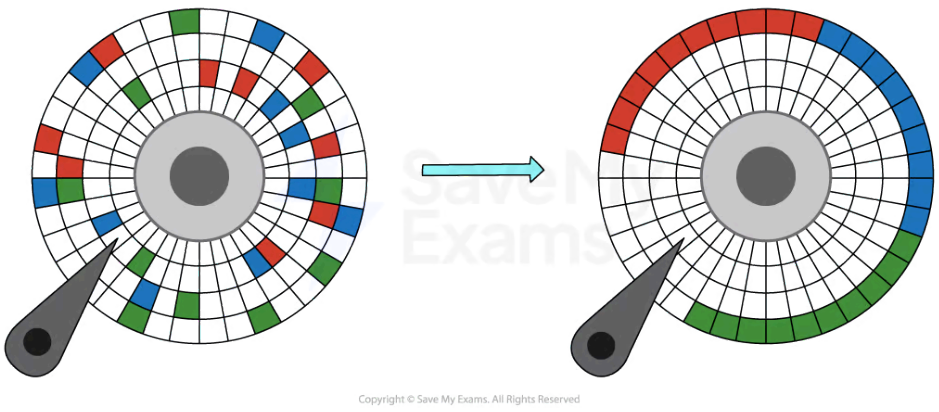
What is Utility Software?

- Utility software is designed to help analyse, configure, optimise, or maintain a computer
- It **supports** the operating system, which is different to application software that performs tasks that benefit end-users

Utility Software	Purpose	Role	Examples/Notes
Disk Defragmentation	To rearrange the files on a hard drive to increase efficiency	By putting files into contiguous blocks and minimizing empty spaces, disk defragmentation speeds up file access and can improve overall system performance	Modern SSDs generally don't require defragmentation
File Management	To organize, search, rename, and relocate files stored on the system	Includes creating, deleting, moving, and renaming files and folders	Windows Explorer, macOS Finder
Device Driver	To provide an interface between the hardware and the operating systems	Ensures that the OS and programs can communicate with the hardware without needing to know the hardware's precise details	The OS usually manages drivers, but users may sometimes need to update them
System Cleanup	To free up space on the system by removing unnecessary files and data	System cleanup utilities remove temporary files, system cache, unused applications, and other "junk" that can slow down the system	CCleaner, Disk Cleanup on Windows

Security	To protect the computer system from various threats like viruses, malware, and spyware	Monitors the system and controls the computer's activities to protect it from threats	Norton, McAfee, Windows Defender

Your notes



Disk defragmentation process

Worked Example

Operating systems usually come with utility software pre-installed. Give two examples of utility software, explaining the purpose of both.

[4]

Answer:

Example answer that gets full marks:

Disk defragmentation is a utility that will better organise files on the hard disk so that the operating system can access them more efficiently. Better organised files will lead to a smoother operation of the system.

File encryption software enables users to transfer sensitive data files over a network securely. A simple encryption utility will request a password from the user and scramble the file's contents. The file contents will only be reassembled in the correct order if the receiver knows the password.

Acceptable answers you could have given instead:

Disk defragmentation rearranges files on a computer's hard disk to make it run more smoothly. File encryption software lets users send private files safely by scrambling the contents, and only someone with valid permission can unscramble them.

Your notes

Open Source & Closed Source Software

- Think of software like a recipe. It's a set of instructions that tell a computer what to do
- **Open Source Software (OSS)** is like a **shared recipe**. Anyone can look at it, change it, or share it with others. It's all about community and collaboration
- **Closed Source Software (CSS)** is like a **secret recipe** that only particular chefs know. People can taste the dish at a restaurant but can't see the exact ingredients or change the recipe. This is how closed-source software works; the instructions are kept secret
- Knowing a little bit about these two types of software can help in understanding what's happening behind the scenes and in making better choices about the software that is used

Definition, Examples, and Typical Usage Scenario

	Definition	Examples	Typical Usage Scenario
Open Source Software	Users can view, modify, and distribute the **source code**	Linux, Apache HTTP Server	Ideal for collaborative projects, customization, transparency
Closed Source Software	The source code is hidden and proprietary.	Microsoft Windows, Adobe Photoshop	Ideal for businesses requiring polished, supported products, **intellectual property** protection.

Benefits and drawbacks to the creator

	Benefits	Drawbacks
Open Source Software	Collaboration, community engagement, faster innovation	Less control, burdened with requests from users
Closed Source Software	Greater control, revenue through sales, IP protection	Slower innovation, full responsibility for updates & flaws

Benefits and drawbacks to the user

	Benefits	Drawbacks
Open Source Software	Often free, customisable, transparent	Might be less user-friendly, compatibility issues, may contain bugs
Closed Source Software	More polished products, professional support, consistency	Costly, less customisable, potential trust issues

Your notes

Worked Example

Imogen installs a compiler for a high-level programming language onto her computer and makes use of an open-source IDE.

State what is meant by the term 'open source software'.

[2]

Answer:

Example answer that gets full marks:

Open Source Software is where the original source code is made freely available and may be redistributed and modified. This means that anyone can view, access, and modify the code.

Worked Example

Give one benefit to Imogen of using an open-source IDE rather than a closed source IDE.

[1]

Answer:

Example answer that gets full marks:

One benefit to using an open-source IDE is the ability to customise the IDE according to her specific needs. This level of customisation is typically not available in closed-source software, where the source code is proprietary and can't be altered by the end user.

Translators

Your notes

Translators

What is a Translator?

- Translators convert source code from a high-level language to a low-level language
- There are three main types of translators
    - Interpreters
    - Compilers
    - Assemblers

Interpreters

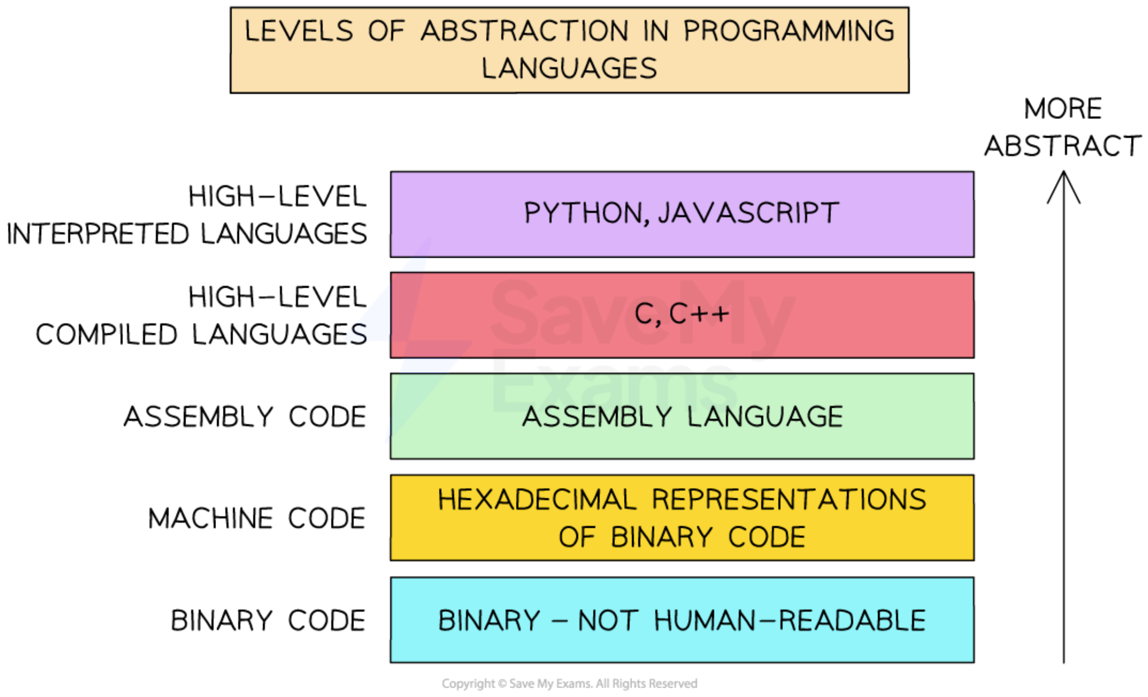
- Interpret source code **line-by-line** and **executes it on the fly**
- Easier to debug, allows incremental testing, and is generally **faster to start execution**
- **Slower execution time** overall and requires the interpreter to be present during the execution

Compilers

- Translates the **entire source code into machine code** at once and then executes it
- **Faster execution time**, no need for the compiler during execution
- **Longer initial compilation time** and can be more challenging to debug

Assemblers

- Assemblers translate **assembly language** into machine code
- Unlike interpreters and compilers that work with high-level languages, assemblers deal with low-level languages
- The diagram below shows examples of programming languages, from low-level to high-level


LEVELS OF ABSTRACTION IN PROGRAMMING LANGUAGES
MORE ABSTRACT
HIGH–LEVEL INTERPRETED LANGUAGES
PYTHON, JAVASCRIPT
HIGH–LEVEL COMPILED LANGUAGES
C, C++
ASSEMBLY CODE
ASSEMBLY LANGUAGE
MACHINE CODE
HEXADECIMAL REPRESENTATIONS OF BINARY CODE
BINARY CODE
BINARY – NOT HUMAN–READABLE
Copyright © Save My Exams. All Rights Reserved

Levels of Abstraction in Programming Languages

Translators used for common languages

Programming Language	Translator
C	Compiler
C++	Compiler
Java	Compiler
Python	Interpreter
JavaScript	Interpreter
Ruby	Interpreter

Swift	Compiler
Assembly Language	Assembler
PHP	Interpreter

Why do different languages have different translators? (non-syllabus content)

- **Level of Abstraction**
  - High-level languages like **Python** or **JavaScript** are **further from machine language** and often use interpreters to allow more **flexibility** and **ease of development**
  - Low-level languages like **C** or **Assembly** are **closer to machine code** and typically use compilers or assemblers for **efficiency**
- **Execution Model**
  - Languages like **C** and **C++** are compiled into machine code specific to a **target platform**
  - This allows for optimisations to make the code run more **efficiently**
  - But the code is generally **not portable** between different platforms
  - Languages like **Python** and **JavaScript** are interpreted, meaning they are translated line-by-line at runtime **on any platform**
  - This provides **greater portability**, as the same code can run on different platforms
  - This may **sacrifice some performance** compared to compiled code
- **Development Paradigm**
  - Interpreted languages allow determining variable types at runtime, which makes development faster
    - e.g. `var x = 5` (Interpreter will calculate `x` to be a number at runtime)
  - Compiled languages enforce stricter type-checking at compile time, which requires additional work across the project
    - e.g. `String var name = 'Michael';` (Compiler will demand that `name` has a data type before compiling)

Your notes

Worked Example

The program below is written in assembly code using the Little Man Computer instruction set. It is supposed to take in two numbers and output the higher.

```
INP
STA NUMA
INP
STA NUMB
SUB NUMA
BRP NOTA
LDA NUMB
BRA QUIT
NOTA LDA NUMA
QUIT OUT
HLT
NUMA DAT
NUMB DAT
```

State what type of translator program would be needed to convert the code above into machine code.

[1]

Answer:

Answer that gets full marks:

Assembler.

Stages of Compilation

Your notes

Stages of Compilation

What is Compilation?

- **Compilation** is a process that translates a program written in a high-level programming language into machine code
- Only machine code can be executed by a computer
- There are four stages involved in this process:
    - Lexical Analysis
    - Syntax Analysis
    - Code Generation
    - Optimisation

Lexical Analysis

- Lexical analysis means studying the **words or vocabulary** of a language
- This stage involves identifying lexical 'tokens' in the code
- Tokens represent small meaningful units in the programming language, such as:
    - Keywords
        - var, const, function, for, while, if
    - Identifiers
        - Variable names, function names
    - Operators
        - '+', '++', '-', '*'
    - Separators
        - ',' ';', '{', '}', '(', ')'
- During this stage, unnecessary elements like comments and whitespace are ignored
- For example, if the following code is being compiled:

```
var x = function(x,y) {
```

```
if(x>2) {

return x*y;

}

return x+y;
}
```

- The result of lexical analysis is a **token table**

	Token	Type
1	var	Keyword
2	x	Identifier
3	=	Operator
4	function	Keyword
5	(	Separator
6	x	Identifier
7	,	Separators
8	y	Identifier
9	)	Separator
10	{	Separator
11	return	Keyword
12	x	Identifier
13	*	Operator

Your notes

Your notes

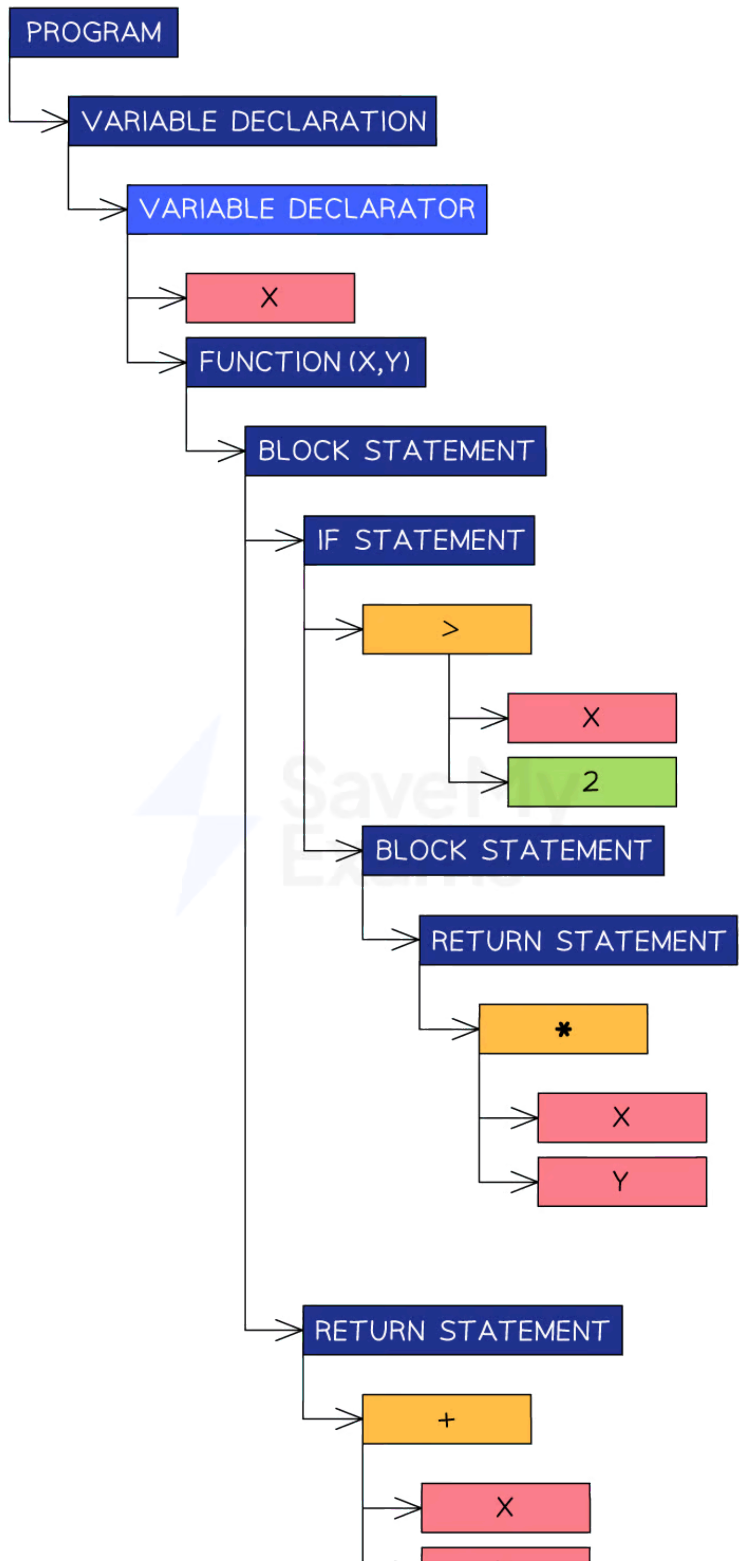
14	y	Identifier
15	;	Separator
16	}	Separator

Syntax Analysis

- Now that tokens have been identified, syntax analysis makes sure they all **adhere to the syntax rules** of the programming language
- A symbol, e.g. '$' could be a valid token but not a valid character according to particular programming languages
- The dollar symbol would be flagged as breaking the syntax rules
- Other syntax errors programmers commonly make include **mismatched parentheses** or **missing semicolons**
- If the code passes the syntax analysis, the compiler can create an **Abstract Syntax Tree** (AST)
- An AST is a graph-based representation of the code being compiled
- An AST is an efficient way to represent the code for the next step

Example abstract syntax tree

- For the same code as above, the following abstract syntax tree can be created


PROGRAM
VARIABLE DECLARATION
VARIABLE DECLARATOR
X
FUNCTION (X,Y)
BLOCK STATEMENT
IF STATEMENT
>
X
2
BLOCK STATEMENT
RETURN STATEMENT
*
X
Y
RETURN STATEMENT
+
X

Abstract syntax tree

Code Generation

- This step takes the AST and traverses it to generate **object code** that can be executed by the computer

Optimisation

- This step modifies the code to make it more efficient without changing its functionality
- This is important to attempt because it reduces the memory required to run the code, which leads to faster execution
- A common optimisation action is removing duplicate code
- If an 'add' function is written twice in the source code, a sophisticated compiler will notice this and include it only once in the object code

Worked Example

Imogen is writing application software for her company and is ready to compile it.

```
const celsius = (fahrenheit) => {
  return (5/9) * (fahrenheit-32);
}
```

Referring to the example above, explain what happens during Lexical Analysis.

[2]

How to answer this question:

- Recall the purpose of lexical analysis and what it aims to produce
- Recall the different types of tokens that can be identified in the code
- Use examples from the code block to write your answer

Answer:

Answer that gets full marks:

'Lexical analysis' breaks the code into tokens, ignoring whitespace and comments. Tokens are identified by their type:

- keyword: 'return'
- operator: '*', '-'

- identifier: celsius, fahrenheit
- delimiter: ';', '(', ')' etc

When all tokens have been identified, a token table is produced for the next step in the compilation.

Acceptable answers you could have given instead:

The compiler will look at all the lexical tokens in the code e.g. 'fahrenheit' is an identifier, 'return' is a keyword. All the tokens are placed into a tokens table for the next step.

Worked Example

State the name of the stage of compilation that directly follows Lexical Analysis.

[1]

Answer:

Answer that gets full marks:

Syntax analysis.

Your notes

Libraries, Linkers & Loaders

What are code libraries?

- **Definition**: A code library is a collection of pre-written code, classes, procedures, scripts, configurations, and more. They are packaged together to allow developers to perform common tasks without having to write code from scratch.

How are libraries used?

- **Inclusion**: Libraries can be included in your code, providing functionalities that are already implemented
- **Standard Libraries**: Most programming languages come with standard libraries that offer basic functionalities
- **Third-Party Libraries**: Developers also have access to libraries created by others, often shared through package managers

Benefits of using libraries

- **Efficiency**: Saves time and effort, as you don't need to write everything from scratch
- **Reliability**: Often tested and optimized, reducing the chance of errors
- **Reusability**: The same library can be used in different parts of the project or in different projects
- **Community Support**: Popular libraries often have strong community support and documentation

Drawbacks of using libraries

- **Dependency Issues**: Relying on a third-party library can lead to problems if the library is discontinued or not maintained
- **Compatibility**: There might be compatibility issues with different versions of the library or the system you are working on
- **Overhead**: Using a large library for a small task can add unnecessary complexity and size to the application

How libraries are used to write programs

- **Practical Application**: Regularly using libraries can enhance programming skills and enable the development of more complex and efficient programs

- **Best Practices**: Knowing when and how to use libraries properly is key to maximizing their benefits

Libraries During Compilation

- **Compilation Process**: Libraries are often compiled separately and then linked to the code during the compilation process
- **Static Linking**: Libraries are combined with the code at compile time, creating a larger executable
- **Dynamic Linking**: Libraries are linked at runtime, allowing for more flexibility but requiring the library to be present during execution

Role of Linkers and Loaders

- **Linkers**: These combine different code files and libraries into a single executable. They resolve references between files, ensuring everything points where it should
- **Loaders**: These are system tools that load executable files into memory so they can be run by the operating system

Worked Example

When Imogen creates programs in a high-level language, she uses libraries.

Explain what a library means, giving one example of when one may be used.

[3]

How to answer this question:

- Provide a clear definition of a library in programming
- Offer an example related to any programming language

Answer:

Example answer that gets full marks:

A library in programming is a collection of pre-written functions and procedures that developers can incorporate to simplify their work. Imogen might use the JavaScript library 'Lodash' when working on complex data manipulation within an application. Lodash offers utility functions that streamline working with arrays, objects, strings, and more, enhancing code readability and efficiency.

Acceptable answers you could have given instead:

A library is a set of reusable code modules in programming. Imogen might use Lodash for a single string manipulation task in her project, which could simplify that task but might be considered overkill if the rest of the library's functionalities are not utilised.

Worked Example

Describe one advantage of the use of library files to programmers.

[2]

How to answer this question:

- Describe one clear and specific advantage of using library files in programming
- Provide an example to highlight the benefit.

Answer:

Example answer that gets full marks:

One advantage of using library files is that they save significant development time by providing pre-written and well-tested code functions, such as data manipulation functions in the Lodash library.

Acceptable answers you could have given instead:

Using library files can standardize development, making code more readable and maintainable, but it may introduce unnecessary complexity if the library is too extensive for the task.

Worked Example

Describe one disadvantage of the use of library files to programmers.

[2]

How to answer this question:

- Describe one specific disadvantage of using library files in programming.

Answer:

Example answer that gets full marks:
One disadvantage of library files is that they can add unnecessary complexity, mainly if an extensive library is used for a minor functionality, like using Lodash for just one function.

Acceptable answers you could have given instead:

Using library files might lead to dependency issues, potentially causing problems if the library becomes unsupported or incompatible.

Your notes

Worked Example

Explain how linkers are used during the compilation process.

[3]

How to answer this question:

- Explain the role of linkers in the compilation process, focusing on their main function.

Answer:

Example answer that gets full marks:

Linkers are used to combine object files and libraries into a single executable file. They resolve symbols between these files and link them together, enabling separate compilation and creating a final program that can be run by the system.

Acceptable answers you could have given instead:

Linkers connect object files into a final executable, allowing for modular programming and utilisation of reusable code, but may complicate dependency management.