Explain why a programmer chooses a high-level language for a general-purpose application and assembly language for a device driver. (6)

Create completed tables like the ones below as a plan before writing the explanation.

Feature	High-Level Language	Assembly Language
Abstraction Level		
Portability		
Ease of Programming		
Development Time		
Readability		
Application Type		
Debugging		
Hardware Access		

Language Type	Advantages	Disadvantages
High-Level Language		
Low-Level Language		

ANSWERS

For long answer questions, begin with definition. In this case, define high-level language and assembly language.

Then take a point and state it. Then explain it and link it to the context. Use the words from the question. If the question is asking for advantages and disadvantages make sure you make it clear. If the question is asking for comparisons, make sure you make the same point for each.

Good linking phrases to ensure explanation include…'because….', 'this means that', 'which means that', 'so that'

For example:

A high-level programming language is one written using English-like words and syntax that is easy for a human to read and understand. Assembly language uses mnemonics which are less easy to understand and harder to learn. (Definitions)

A programmer would use a high-level language to program a general-purpose app (Link to question) because it would be faster than using assembly language (Point). This is because (Explanation) it will have many built-in libraries and sub-programs for common tasks.
They would use assembly language to code a device driver (link to question) because it will produce shorter, more efficient code and gives the programmer direct access to the hardware (Point). This means that (Explanation) the programmer can directly work with the peripheral device concerned.

Feature	High-Level Language	Assembly Language
Abstraction Level	Provides a high level of abstraction, closer to human language	Assembly language is low-level, directly interacting with the hardware. It uses mnemonic codes for instructions.
Portability	High-level languages are machine-independent. Code written in one high-level language can run on different platforms.	Assembly language programs are machine-specific. They are tied to a particular processor. They are not portable without changes.
Ease of Programming	High-level languages are easier to learn and use. They offer built-in functions, libraries, and abstractions for common tasks.	Assembly language requires in-depth knowledge of the hardware. Programmers must manage memory, registers, and errors manually.
Development Time	Faster development time due to simpler syntax and in-built functions and libraries	Slower development time due to more detailed coding
Readability	High-level code is readable by other programmers. It uses meaningful variable names and structured logic.	Assembly code can be harder to understand. It lacks descriptive names and relies on short mnemonics.
Application Type	High-level languages are ideal for general-purpose applications, web development, databases, and business software.	Assembly language is suited for device drivers, real-time systems, and embedded programming where efficiency is critical.
Debugging	High-level languages provide debugging tools,and error messages. Debugging is easier.	Assembly debugging involves manual inspection. It's challenging and time-consuming.
Hardware Access	High-level languages abstract hardware details. Direct hardware access is limited.	Assembly language allows direct manipulation of hardware components. It's essential for low-level tasks like I/O control.

Language Type	Advantages	Disadvantages
High-Level Language	- Easier to learn and use - Faster development time - More portable - More readable and maintainable - Multiple libraries and in-built sub programs save time and require less skills	- Less efficient than low-level languages - May require additional resources to run - Limited control over hardware
Low-Level Language	- More control over hardware - More efficient for specific tasks - Smaller program size - Direct hardware interaction	Difficult to learn and use - Time-consuming development process - Not portable - More prone to errors - Requires in-depth understanding of hardware