1.2.2 Translators

1. Before code can be executed, a translator must be used.

a) State the purpose of a translator.

[1 mark]

- **To convert high-level (or assembly) code to low level/machine code**

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

[4 marks]

- **Compiler translates code all at once / before it is executed**
- **Interpreter translates code line by line / during runtime**

- **Compiler produces executable file for reuse // Doesn't need to be translated each time it is run**
- **Interpreter needs to re-translate every time program is run**

- **Compiler lists all errors // does not produce executable if there are any errors**
- **Interpreter stops execution at the first error**

- **Compiler programs do not need source code to be distributed with a compiled executable**
- **The source code must be available to the interpreter to run**

c) Describe the role of an assembler.

[2 marks]

- **A program that translates assembly code**
- **… into machine / object code**

2. A coding team a looking at writing a new closed source application that converts audio, image and video files from one format to another. They intend to sell copies of the program when it is complete.

They investigate three programming languages they could use, including:

- C++, which is compiled to machine code
- Java, which compiles to an intermediate code that then runs off a virtual machine
- JavaScript, which runs from an interpreter in a web browser.

Discuss the benefits and drawbacks of the three options above and justify which option you would recommend.

[9 marks]

Knowledge

Java

- **A single version of the Java program can run on different platforms that have a suitable Java Virtual Machine (JVM).**
- **Programs running through a VM may be slower than equivalent native compiled programs.**

C++

- **Multiple versions of the code will need to be maintained for different architectures…**
- **…however there may be minimal differences between them, and then just need compiling with different compilers.**
- **Native compiled code will generally execute faster than interpreted or VM-based alternatives.**

JavaScript

- **Interpreted execution is likely to be slower than compiled native code.**
- **Can run in a web browser on many different platforms, provided the browser supports the required JavaScript.**

Application

Java

- **Multiple devices can include devices other than PCs (i.e. phones, tablets).**
- **People with unusual operating systems or architectures would have access to the application.**
- **It makes commercial sense to sell to as wide an audience as possible.**
- **The speed reduction compared to compiled code will likely be noticeable with such a processor intensive task.**
- **As running on a VM coders will have limited (if any) access to some of the low level features (e.g. access to the GPU) which can optimise the program.**
- **Intermediate code is used helping protect intellectual property.**

C++

- **Some less used architectures may not be developed for as not commercially viable.**

- Compiled code will run quicker than the other options. This is likely to be noticeable given the nature of the task.
- Easier to get access to lower level features (such as GPU access).
- Compiled code is not human readable helping to preserve intellectual property

JavaScript

- Most people have web browsers so by far most compatible option (don't even need VM).
- The slow speed may be frustrating…
- …though as no user interaction is needed this may be a trade off worth making.
- Client-side JavaScript source code is generally accessible to the user, making it easier for the code to be inspected or copied.

Evaluation

- Any recommendation can receive full credit if it is supported by a balanced comparison and applied to the scenario.