**Tutorial 01**

M. V. Amarasinghe - 24981

# Question 1

Programming languages serve as a means of communication between humans and computers. Since computers can only process instructions in binary, programming languages provide a higher-level and more human-readable alternative. By writing instructions in a programming language, we can convey our intentions and desired tasks to the computer. These instructions are then translated into binary code using interpreters or compilers, allowing the computer to execute the desired operations accurately.

# Question 2

## a)

Source code is the code written by programmers using high-level programming languages. It is human-readable and consists of letters, numbers, and symbols that are easily understood by humans. Source code needs to be compiled or interpreted before it can be executed by the computer. It allows for easier debugging and maintenance.

On the other hand, machine code, also known as machine language, is a lower-level language consisting of binary code represented by 1s and 0s. It is not human-readable and can be challenging for programmers to work with directly. Machine code is specific to each computer architecture and is typically generated by compilers or assemblers from the source code.

## b)

High-level languages are designed to be user-friendly and easily understood by humans. They use words, numbers, and symbols that resemble everyday language, making programming more intuitive. High-level languages provide abstraction and built-in functionality, allowing programmers to focus on problem-solving rather than dealing with hardware-specific details. They are portable and can run on different systems with minimal modifications.

However, low-level languages are closer to the computer's hardware and offer more direct control. They have a smaller set of commands that are closely tied to the computer's architecture. Low-level languages, such as machine code and assembly language, use binary that are harder for humans to read and understand without a translation tool. They consider efficiency over readability.

## c)

A compiler takes the entire source code as input and translates it into machine code in a one-time process. The resulting output is an executable file that can be directly executed by the hardware. The generated machine code is independent of the compiler and can be run on compatible hardware platforms.

An interpreter executes code line by line. It analyzes and executes each line of code during runtime, providing instant feedback on errors and issues. Unlike a compiler, an interpreter does not produce intermediate machine code. Instead, it directly interprets and executes the code in a more interactive and incremental manner.

## d)

Structured programming focuses on modular decomposition and control flow, while object-oriented programming emphasizes the organization of code around objects and their interactions. Structured languages are suitable for smaller programs, while object-oriented languages excel in modeling complex systems and real-world scenarios.

## e)

C is a procedural language known for its simplicity and efficiency. It lacks built-in support for object-oriented programming features and requires manual memory management. C is commonly used in system programming and low-level tasks.

C++ is an extension of the C language that introduces additional features, particularly in object-oriented programming. It includes the concept of classes, allowing for the organization of data and behavior into objects. C++ has a larger codebase and provides a higher level of abstraction compared to C. It is widely used in application development, game development, and other areas where object-oriented programming is beneficial.

## f)

C++ and Java are both widely used programming languages, but they have different characteristics and purposes. C++ is an extension of the C language, emphasizing object-oriented programming and used in application and game development. Java is a platform-independent language with a focus on enterprise applications and web development, offering a vast standard library.

## g)

A syntax error occurs when the code violates the rules or syntax of the programming language. It is a mistake in the structure of the code, such as missing symbols, incorrect spelling of a command, or improper use of language constructs. Syntax errors are typically detected by the compiler or interpreter during the compilation or execution process. When a syntax error is encountered, the code cannot be executed until the error is fixed.

A logical error occurs when the code does not produce the expected result or behaves incorrectly. It is a mistake in the algorithm or the logic of the program. Logical errors do not violate syntax rules and, therefore, do not cause the code to fail to compile or execute. Identifying logical errors can be more challenging as they can lead to unexpected and incorrect results. Debugging and careful analysis of the code's logic are required to identify and rectify logical errors.