2.Define the following terms as used in C programming:

a)Compiler: is a program that translates a programming languages source code into another programming language.

b)source code: is a group of instructions a programmer writes using computer programming languages.

c)Object code: set of instructions understood by a computer at the lowest hardware level.

d)Linkers: program that takes 1 or more object files and combines them into a single executable file, library file or another object file

3.Explain the differences between a compiler and interpreter

|  |  |
| --- | --- |
| Compiler | Interpreter |
| * Translates high level code into machine code | * Executes high level code directly without compiling |
| * Produces a separate executable file | * Executes code line by line in real time |
| * Reports errors at complete time | * Reports errors at run time |
| * Has faster execution of translated code | * Slower execution due to interpretation |
| * Requires more memory to compile the code | * Requires less memory as code is not compiled |
| * Generates machine code which can be optimized for performance | * Performance is generally slower due to interpretation |

5.List all main categories of operators available in C programming and specific operators in each category

1. Arithmetic operators: perform common math operations.

Specific operator; addition(+), subtraction(-), multiplication(\*), division(/),

Modulus(%), increment(++), decrement (--)

1. Assignment operators: assign values to variables

Specific operator; =, +=, -=, \*= , /=, %=, &=, |=, ^=, >>=, <<=

1. Comparison operators: used to compare 2 variables

Specific operator: Equal to(==), not equal(!=), greater than(>), less than(<), greater than or equal to(>=), less than or equal to(<=)

1. Logical operators: used to determine the logic between variables or values

Specific operator: logical and(&&), logical or(||), logical not(!)

1. Bitwise operator: perform bit-level operations on the operands

Specific operator: Bitwise AND (&) ,Bitwise OR(|), Bitwise exclusive OR(^), Bitwise complement(~), shift left(<<), shift right(>>)

1. Special Operators: &, \*, size of ()

3.Using an example ie. A program to add 2 numbers explain the compilation process.

Pre-processing: The pre-processor handles directives starting with #.It might include files specified with # include and perform macro replacements.

Compiling: The compiler translates the pre-processed codes into assembly code

Assembling: The assembler converts the assembly code into machine code specific to the target architecture.

Linking: linker combines the object code with other object codes (ie libraries) to produce an executable file. It resolves addresses and creates the final binary.

Execution: The user runs the compiled program,and it adds the numbers during runtime, displaying the result as specified in the printf statement.