**Question:**

1. Write short note on the following;
2. Assembler
3. Linkers
4. Compiler
5. Interpreter
6. Debugging
7. Syntax and semantics
8. Parsing
9. Develop and Algorithm (pseudocode and flowchart) for the following;
10. Checking and validating students into an examination hall
11. To calculate the area of a rectangle of length L, and width, N.

**Answer:**

**A. Assembler**

An assembler is a program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations. Some people call these instructions assembler language and others use the term assembly language

The programmer write programs using a sequence of assembler instructions. The sequence of assembly instructions, known as the source code or source program, can be specified to the assembler program when that program is started.

The assembler program takes each program statement in the source program and generates a corresponding bit stream or pattern (a series of 0's and 1's of a given length). The output of the assembler program is called the object code or object program relative to the input source program. The sequence of 0's and 1's that constitute the object program is sometimes called machine code. The object program can then be executed whenever desired.

1. **Linkers**

A linker or link editor is a computer system program that takes one or more object files (generated by a compiler or an assembler) and combines them into a single executable file, library file, or another "object" file. Object files and static libraries are assembled into a new library or executable file.

1. **Compiler**

A compiler is a program that translates a source program written in some high-level programming language (such as Java) into machine code for some computer architecture (such as the Intel Pentium architecture). The generated machine code can be later executed many times against different data each time.

Programming languages that use compilers include Java, C++, and C.

1. **Interpreter**

An interpreter is a computer program that reads an executable source program written in a high-level programming language as well as data for this program, and it runs the program against the data to produce some results. One example is the Unix shell interpreter, which runs operating system commands interactively.

The interpreter transforms the high-level program into an intermediate language that it then executes, or it could parse the high-level source code and then performs the commands directly, which is done line by line or statement by statement.

The interpreter reads each statement of code and then converts or executes it directly. In contrast, an assembler or a compiler converts a high-level source code into native (compiled) code that can be executed directly by the operating system.

Programming languages that use interpreters include Python, Ruby, and JavaScript.

1. **Debugging**

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.

In software development, the debugging process begins when a developer locates a code error in a computer program and is able to reproduce it. Debugging is part of the software testing process and is an integral part of the entire software development lifecycle.

In hardware development, the debugging process typically looks for hardware components that are not installed or configured correctly. For example, an engineer might run a JTAG connection test to debug connections on an integrated circuit.

A debugger is a software tool that can help the software development process by identifying coding errors at various stages of the operating system or application development.

1. **Syntax vs. Semantics**

In defining or specifying a programming language, we generally distinguish between syntax and semantics. The **syntax** of a programming language describes which strings of characters comprise a valid program. The **semantics** of a programming language describes what syntactically valid programs mean, what they do. In the larger world of linguistics, syntax is about the form of language, semantics about meaning. Linguistic giant Noam Chomsky gave this nice sentence:

In programming languages our expectations of what certain symbols should mean, like "+" or "if" may make us lose sight of the fact that both syntax and semantics really need to be defined to define a language. However, there are places where this becomes pretty clear. For example, in a C/C++ program the following is syntactically correct:

int x;

x = 2^3;

1. **Parsing**

Parsing in computer science, is where a string of commands where a program is separated into more easily processed components, which are analyzed for correct syntax and then attached to tags that define each component. The computer can then process each program chunk and transform it into machine language.

To parse is to break up a sentence or group of words into separate components, including the definition of each part's function or form. The technical definition implies the same concept.

Parsing is used in all high-level programming languages. Languages like C++ and Java are parsed by their respective compilers before being transformed into executable machine code. Scripting languages, like PHP and Perl, are parsed by a web server, allowing the correct HTML to be sent to a browser.

2. Checking and Validating students into an examination hall
3. Calculate the Area of a Rectangle of Length, L and width, N

Program Algorithm: Area of Rectangle

BEGIN

Get Length, L and Width, N of the rectangle.

Do Area = L\*W

Output Area

END

INPUT L, N

Area = L\*N

OUTPUT Area