1. **Discuss various types of computer languages.**

Ans. In order to communicate with the computer user also needs to have a language that should be understood by the computer. For this purpose, different languages are developed for performing different types of work on the computer. Basically, languages are divided into two categories according to their interpretation.

- ❖ Low Level Languages.
- ❖ High Level Languages.

- **Low Level Languages** : Low level computer languages are machine codes or close to it. Computer cannot understand instructions given in high level languages or in English. It can only understand and execute instructions given in the form of machine language i.e. language of 0 and 1. There are two types of low level languages:
  - ➢ **Machine Language** : It is the lowest and most elementary level of Programming language and was the first type of programming language to be developed. Machine Language is basically the only language which computer can understand. In fact, a manufacturer designs a computer to obey just one language, its machine code, which is represented inside the computer by a string of binary digits (bits) 0 and 1. The symbol 0 stands for the absence of electric pulse and 1 for the presence of an electric pulse . Since a computer is capable of recognizing electric signals, therefore, it understands machine Language.

    Advantages of Machine Language

    It makes fast and efficient use of the computer.

    It requires no translator to translate the code i.e. directly understood by the computer

    Disadvantages of Machine Language

    All operation codes have to be remembered

    These languages are machine dependent i.e. a particular Machine language can be used on only one type of computer.
  - ➢ **Assembly Language** : It was developed to overcome some of the many inconveniences of machine language. This is another low level but a very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0's and 1's. These alphanumeric symbols will be known as mnemonic codes and can have maximum up to 5 letter combination e.g. ADD for addition, SUB for subtraction, START, LABEL etc. Because of this feature it is also known as "Symbolic Programming Language". This language is also very difficult and needs a lot of practice to master it because very small english support is given to this language. The language mainly helps in compiler orientations. The instructions of the assembly language will also be converted to machine codes by language translator to be executed by the computer.

    Advantages of Assembly Language

    It is easier to understand and use as compared to machine language.

    It is easy to locate and correct errors.

    It is modified easily

    Disadvantages of Assembly Language

    Like machine language it is also machine dependent. Since it is machine dependent therefore programmer should have the knowledge of the hardware also.

- **High Level Languages** : High level computer languages give formats close to english language and the purpose of developing high level languages is to enable people to write programs easily and in their own native language environment (english). High-level languages are basically symbolic languages that use english words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high level language is translated into many machine language instructions thus showing one-to-many translation Types of High Level Languages. Many languages have been developed for achieving different variety of tasks, some are fairly specialized others are quite general purpose. These are categorized according to their use as

  (a) **Algebraic Formula-Type Processing**. These languages are oriented towards the computational procedures for solving mathematical and statistical problem.
  Examples are BASIC (Beginners All Purpose Symbolic Instruction Code), FORTRAN (Formula Translation), PL/I (Programming Language, Version 1), ALGOL (Algorithmic Language), APL (A Programming Language).

  (b) **Business Data Processing**: These languages emphasize their capabilities for maintaining data processing procedures and files handling problems. Examples are: COBOL (Common Business Oriented Language), RPG (Report Program Generator.

  (c) **String and List Processing** : These are used for string manipulation including search for patterns, inserting and deleting characters. Examples are: LISP (List Processing), Prolog (Program in Logic).

  (d) **Object Oriented Programming Language** : In OOP, the computer program is divided into objects. Examples are: C++ , Java.

  (e) **Visual programming language**: These are designed for building windows-based applications. Examples are: Visual Basic, Visual Java, Visual C.

2. **What is C?**

Ans. C is a programming language developed at AT & T"s Bell Laboratories of USA in 1972. It was designed and written by Dennis Ritche. Dennis Ritchie is known as the founder of C language. It was developed to overcome the problems of previous languages such as B, BCPL etc. Initially, C language was developed to be used in UNIX operating system.

Features of C

- Portability or machine independent
- Sound and versatile language
- Fast program execution.
- An extendible language.
- Tends to be a structured language.

3. What are macros? What are its advantages and disadvantages?

Ans: Macros are abbreviations for lengthy and frequently used statements. When a macro is called the entire code is substituted by a single line though the macro definition is of several lines. The advantage of macro is that it reduces the time taken for control transfer as in case of function. The disadvantage of it is here the entire code is substituted so the program becomes lengthy if a macro is called several times.

4. What is a function?

Ans: A large program is subdivided into a number of smaller programs or subprograms. Each subprogram specifies one or more actions to be performed for the larger program. Such sub programs are called functions.

5. Distinguish between syntax vs logical error?

Ans: Syntax Error 1-These involves validation of syntax of language. 2-compiler prints diagnostic message.
Logical Error 1-logical error are caused by an incorrect algorithm or by a statement mistyped in such a way that it doesn't violet syntax of language. 2-difficult to find.

6. What is preincrement and post increment?

Ans: ++n (pre increment) increments n before its value is used in an assignment operation or any expression containing it. n++ (post increment) does increment after the value of n is used.

7. What are the various types of operators?

Ans. C supports the following types of operators:
Unary Plus +
Unary Minus –
Increment ++
Decrement –
Binary Arithmetic +,-,*,/,%
Relational ,<=,>=,==,!=
Logical &&,||,!
Bitwise &,|,^,<>
Assignment =
Shorthand assignment +=, -=,*=,/=,%=
Ternary ?:
Special Sizeof(), *. ->

8. Write any two preprocessor directives in C ?

Ans. Preprocessor directive instructions are used to instruct the preprocessor to expand/translate the source program before compilation.
The two types of preprocessor directives are
#define to define macro,
#include to include library files.

9. What is the variable? Illustrate with an example.

Ans. Variable is a named storage area used to assign a name to a value. To declare a variable, we need to specify the data type of the value and the variable name: data type variable_name. - Example int reg; float avg.

10. What is the use of #define pre-processor ?

Ans. The #define preprocessor directive is used to define a
Global constant #define PI 3.14
Macro #define big(a,b) (a+b)
When a compiler reads the macro name, it replaces the macro name by its expansion. When it reads the constant label, it replaces the label by its value.

11. List out various Input & output statements in C ?

Ans. The input & output statements are classified into formatted & unformatted I/O.
Formatted I/O : User can able to design/format the output
Unformatted I/O: doesn't allow the users to design the output

Type	Input	Output
Formatted	Scanf()	Printf()
Unformatted	Getch(), getche() getchar() gets()	putch(), putchar() puts()

12. List the header files in 'C' language.

Ans. <stdio.h> contains standard I/O functions
<stdlib.h> contains general utility functions
<string.h> contains string manipulation functions
<math.h> contains mathematical functions

13. Is it better to use a macro or a function?

Ans.

- Macros are more efficient (and faster) than function, because their corresponding code is inserted directly at the point where the macro is called.
- There is no overhead involved in using a macro like there is in placing a call to a function. However, macros are generally small and cannot handle large, complex coding constructs.
- In cases where large, complex constructs are to handled, functions are more suited, additionally; macros are expanded inline, which means that the code is replicated for each occurrence of a macro.

14. List any five-library functions related to mathematical functions.

Ans. exp(x), sqrt(x), log(x), pow(x,y), sin(x).

15. What is a function in C? Explain the steps in writing a function in C program with Example.

Ans: A function is a block of statements that performs a specific task.

Syntax:
Return_type function_name (argument list)
{
Set of statements – Block of code
}
Example:

```
#include<stdio.h>
 int addition(int num1, int num2);
 int main()
{
 int var1, var2;
 printf("Enter number 1: ");
scanf("%d",&var1);
 printf("Enter number 2: ");
 scanf("%d",&var2);
 int res = addition(var1, var2);
 printf ("Output: %d", res);
 }
int addition(int num1, int num2)
 {
 int sum;
 sum = num1+num2;
 return(sum);
 }
```

16. Explain the iterative statements in C language with examples.

Ans. We have two types of looping statement

- One in which condition is tested before called entry control loop (for loop & while loop).
- The other in which condition is checked at exit called exit controlled loop (do while loop).

Step to be followed while working with loops:

- First initialization is done once
- Then condition is evaluated and if it is true then body of loop is executed
- After execution of body the control goes to increment/ decrement part then condition is once again evaluated and if true body of the loop is again executed.
- While loop syntax:
  while(test condition)

{
body of the loop
}

- for loop syntax:
for(initialization; test control; increment/decrement)
{
body of loop
}
- Do While loop syntax: The while loop does not allow body to be executed if test condition is false. The do while is an exit controlled loop and its body is executed at least once.
do
{
body
}while(test condition);

Note: Write any suitable program to illustrate the working of the above loops.

17. Define Switch Statement.

Ans. A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

18. What is the purpose of continue statement?

Ans. The continue statement in C programming works somewhat like the break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

19. What is a function? Why we use functions in C language? Give an example.

Ans: **Function in C:**
A function is a block of code that performs a specific task. It has a name and it is reusable .It can be executed from as many different parts in a program as required, it can also return a value to calling program.
All executable code resides within a function. It takes input, does something with it, then give the answer. A C program consists of one or more functions.
A computer program cannot handle all the tasks by itself. It requests other program like entities called functions in C. We pass information to the function called arguments which specified when the function is called. A function either can return a value or returns nothing. Function is a subprogram that helps reduce coding.

Simple Example of Function in C

```
#include<stdio.h>
 #include int adition (int a, int b);
int main()
{ int z;
 z= addion(10,3);
 printf ("The Result is %d", z);
}
int addition (int a, int b)
{ int r;
 r=a + b;
return (r);
 }
 Output: The Result is 13
```

Why use function:
Basically there are two **reasons** because of which we use functions
1. Writing functions avoids rewriting the same code over and over. For example - if you have a section of code in a program which calculates the area of triangle. Again you want to calculate the area of different triangle then you would not want to write the same code again and again for triangle then you would prefer to jump a "section of code" which calculate the area of the triangle and then jump back to the place where you left off. That section of code is called 'function'.
2. Using function it becomes easier to write a program and keep track of what they are doing. If the operation of a program can be divided into separate activities, and each activity placed in a different function, then each could be written and checked more or less independently. Separating the code into modular functions also makes the program easier to design and understand.

20. Write some properties and advantages of user defined functions in C?

Ans: **Properties of Functions :**

- Every function has a unique name. This name is used to call function from "main()" function.
- A function performs a specific task.
- A function returns a value to the calling program.

Advantages of Functions in C :

- Functions has top down programming model. In this style of programming, the high level logic of the overall problem is solved first while the details of each lower level functions is solved later.
- A C programmer can use function written by others.
- Debugging is easier in function.
- It is easier to understand the logic involved in the program.
- Testing is easier.

21. Explain the need for the following: #include<stdio.h> #include<math.h>.

Ans: The header file <stdio.h> is used to include and link standard input/output functions in a C program.
The header file <math.h> is used to include and link mathematical functions like sqrt( ), fabs( ) and so on in a C program.

22. What are assemblers, compilers and interpreters ?

Ans:
Assembler: An assembler is a computer program or translator which translates an assembly language program into a machine language program.
Compiler: Compiler translates a source program, written in high-level language into machine language. Compiler takes the whole program at the input and check for errors. So, it is fast in the speed. Most of the high level languages are compiled language. One of the feature of the compiler that, it will convert the source program in the object program. So, every time no need of source program.
Interpreter: Interpreter translates each statement of the source program into a sequence of machine level instructions. Interpreters takes the input line by line and check for the errors. So, it is slower than compiler. Interpreter always need the source program every time.

23. State the differences between compiler and interpreter. State the advantages of high level language.

Ans: The following table lists the differences between a compiler and an interpreter.

	Compiler	Interpreter
1	Scans the entire program first and then translates it into machine code	Translates the program line by line
2	Converts the entire program to machine code; when all the syntax errors are removed execution takes place.	Each time the program is executed, every line is checked for syntax error and then converted to equivalent machine code.
3	Slow for debugging	Good for fast debugging
4	Execution time is less	Execution time is more

Advantage of High-level Programming Language:

There are four main advantages of high-level programming languages.
These are:
(i) **Readability**: Programs written in these languages are more readable than assembly and machine language.
(ii) **Portability**: Programs could be run on different machines with little or no change. We can, therefore, exchange software leading to creation of program libraries.
(iii) **Easy debugging**: Errors could be removed (debugged).
(iv) **Easy Software development**: Software could easily be developed. Commands of programming language are similar to natural language (English).

24. Distinguish between break and continue statements.

Ans.

	Break	Continue
1	Keyword 'break' is used	Keyword 'continue' is used
2	It is used to terminate the loop	It is used to continue the next iteration in the loop
3	It is used in switch case and looping statements	It is used in looping statements only
4	In break statement, control transfers outside the loop	In continue statement, control remains in the same loop

25. When is a switch statement better than multiple if statements ?

Ans. A switch statement is generally best to use when you have more than two conditional expressions based on a single variable of numeric type. For instance, rather than the code

```
if (x == 1)
printf("x is equal to one.\n");
else if (x == 2)
 printf("x is equal to two.\n");
 else if (x == 3)
printf("x is equal to three.\n");
```

```
else printf("x is not equal to one, two, or three.\n");
```

the following code is easier to read and maintain:

```
switch (x)
 {
case 1: printf("x is equal to one.\n"); break;
 case 2: printf("x is equal to two.\n"); break;
 case 3: printf("x is equal to three.\n"); break;
default: printf("x is not equal to one, two, or three.\n"); break;
 }
```

26. Is a default case necessary in a switch statement ?

Ans. No, but it is not a bad idea to put default statements in switch statements for error- or logic-checking purposes.

27. Distinguish between while and do-while loop.

Ans.

while	**do-while**
Condition is checked first then statement(s) is executed.	Statement(s) is executed atleast once, thereafter condition is checked.
It might occur statement(s) is executed zero times, If condition is false.	At least once the statement(s) is executed.
No semicolon at the end of while. while(condition)	Semicolon at the end of while. while(condition);
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
while loop is entry controlled loop.	do-while loop is exit controlled loop.
while(condition) { statement(s); }	do { statement(s); } while(condition);