Q1 What are Decision statements? Define if and switch statement, while, for and do while loops, Loop control statements in brief.

Decision statements are used in programming to make decisions based on certain conditions. They allow the program to **execute different code depending on the outcome** of a logical test.

The "if" statement is a type of decision statement that allows the program **to execute certain code only if a certain condition is met**. The syntax for an "if" statement typically includes the keyword "if," followed by the condition to be tested in parentheses, and a block of code to be executed if the condition is true.

The "switch" statement is another type of decision statement that allows the program to choose from multiple options based on the value of a variable. The syntax for a "switch" statement typically includes the keyword "switch," followed by the variable to be tested, and multiple "case" statements for each possible value of the variable.

Loop control statements are used to control the flow of a program during iteration, such as executing a block of code multiple times. The most common loop control statements are "for," "while," and "do-while."

A "for" loop allows the programmer to set a loop counter, specify the number of iterations, and define the increment or decrement of the loop counter at the beginning of the loop.

A "while" loop repeatedly executes a block of code as long as the condition is true.

A "do-while" loop is similar to a "while" loop, but it guarantees that the block of code will be executed at least once before the condition is tested.

Q2) What are jump statements, break, continue, goto statements, define in brief.

Jump statements are a type of control flow statement that allow the program to transfer control to a different location in the code. They include:

The "break" statement, which is used to exit a loop or switch statement prematurely. When the program encounters a "break" statement, it immediately exits the current loop or switch and continues executing the code that follows it.

The "continue" statement, which is used to skip an iteration of a loop. When the program encounters a "continue" statement, it immediately starts the next iteration of the loop, skipping any code that comes after the "continue" statement in the current iteration.

The "goto" statement, which is used to transfer control to a different location in the code identified by a label. The goto statement is not recommended to use in most of the case, because it can make the code hard to read and understand, but it can be useful in certain situations.

Q3) What are One dimensional array and two-dimensional array, declaration and initialization of one-dimensional arrays and two-dimensional arrays.

A one-dimensional array is a type of data structure that **stores a linear list of elements, all of the same data type**. Each element in the array is **identified by an index**, and the elements are stored in contiguous memory locations.

To **declare a one-dimensional array**

```
data_type array_name[size];
```

For example, to **declare an array of integers called "numbers" that can hold 5 elements:**

```
int numbers[5];
```

You can initialize the elements of a one-dimensional array during the declaration using the following syntax:

```
data_type array_name[] = {element1, element2, ..., elementn};
```

For example:

```
int numbers[] = {1, 2, 3, 4, 5};
```

A **two-dimensional array**, **also known as a matrix**, is an array of arrays. Each element in a two-dimensional **array is identified by two indices**, **one for the row and one for the column**.

To **declare a two-dimensional array** :

```
data_type array_name[rows][columns];
```

For example, **to declare a 2x3 array of integers called "matrix"**:

```
int matrix[2][3];
```

You **can initialize the elements of a two-dimensional array** during the declaration using the following syntax:

```
int matrix[][] = {{1, 2, 3}, {4, 5, 6}};
```

To **access the elements of a two-dimensional array**, use two indices, one for the row and column.

```
array_name[row][column];
```

Q4) What are multi-dimensional arrays? Initialization and accessing multi-dimensional arrays.

A **multi-dimensional array** is an array with **more than two dimensions**. The syntax for declaring and accessing elements of a multi-dimensional array is similar to that of a two-dimensional array, but with more indices. For example, **a three-dimensional array is declared** as:

```
data_type array_name[size1][size2][size3];
```

and **can be accessed using**

```
array_name[index1][index2][index3];
```

Q3) What are User defined and built-in Functions? Also Define storage classes, Parameter passing in functions, call by value.

Functions are self-contained blocks of code that can be called by other parts of a program.

User-defined functions are functions that are created by the programmer. They are typically used to perform specific tasks or to encapsulate a specific piece of logic. They can be defined in a program using a specific syntax, which varies depending on the programming language.

Built-in functions are functions that are already defined in a programming language or in a library. They are typically used to perform common tasks such as input/output operations, mathematical calculations, and string manipulation.

Storage classes are used to define the scope, lifetime and linkage of variables in a program. They define how a variable can be accessed and where it is stored in memory. **The most common storage classes are:**

Automatic storage class, which is the default storage class for variables defined within a function. These variables are created when the function is called and destroyed when the function exits.

Static storage class, which defines a variable with a scope that is limited to the file or block in which it is defined, but with a lifetime that extends throughout the entire execution of the program.

External storage class, which defines a variable that is visible to all the files in a program, and it's stored in a global memory.

Parameter passing in functions refers to the way in which data is passed to a function when it is called. The two most common methods of parameter passing are:

Call by value, in which the value of the argument is passed to the function. The function operates on a copy of the argument, and any changes made to the argument within the function do not affect the original value.

Call by reference, in which a reference to the memory location of the argument is passed to the function. The function operates directly on the memory location, and any changes made to the argument within the function are reflected in the original value.

Q5) How to pass arrays to functions? What is idea of call by reference in Recursion.

arrays can be passed to a function in two ways:

By passing the array name, which is equivalent to passing a pointer to the first element of the array. This method is known as "call by reference" because the function operates directly on the memory locations of the original array. For example:

```
void print_array(int array[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
}
```

By passing the size of the array and the address of the first element of the array. This method is **also known as "call by reference"** because the function operates directly on the memory locations of the original array. For example:

```
void print_array(int *array, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
}
```

In recursion, the idea of **call by reference means that a function calls itself**, passing a reference to the current state of the problem. As the function calls itself with different arguments, it modifies the state of the problem, and the modifications are reflected in the original state.

For example, **a recursive function to find the factorial of a number** could be written as:

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n-1);
    }
}
```

Here the function calls itself with the value of "n-1" as the argument, and the value of the argument decreases until it reaches 0. Each call of the function modifies the state of the problem, and the final result is returned to the original call.

Q6) What are Strings? Define Arrays of characters, variable length character strings, inputting character strings, character library functions, string handling functions.

Strings are sequences of characters that are used to represent text in a programming language. In C, **strings are represented by arrays of characters**, with the **last character being the null character** ('\0').

An **array of characters** is a data type in C that can store a sequence of characters. It is declared with the syntax:

```
char array_name[size];
```

For example**, to declare an array of characters called "name" that can hold 10 characters**:

```
char name[10];
```

A **variable length character string** is a string **where the size of the array is not fixed** and **can change during the execution of the program**. In C, variable length strings are usually implemented using pointers to characters.

Inputting character strings in C can be done using the scanf or gets function. The **scanf function is typically used to input a string with spaces, and it uses the format specifier %s:**

```
char string[20];
scanf("%s", string);
```

The **gets function is typically used to input a string with spaces**, and it takes a pointer to a character array as an argument:

```
char string[20];
gets(string);
```

C has a **set of character library functions**, such **as isalpha, isdigit, toupper, tolower** etc. These functions **perform various operations on individual characters**, such as checking if a character is a letter, a digit or converting it to uppercase or lowercase.

C also has **a set of string handling functions**, such as **strlen, strcpy, strcat, strcmp** etc. These functions **perform various operations on strings**, such as finding the length of a string, copying a string, concatenating two strings or comparing two strings. **These functions are defined in the <string.h> header file.**
