Q.Explain brief history of 'C' programming.

ANS:

- **C programming language was developed by Dennis Ritchie in 1972 at Bell Laboratories.**
- **It was developed for writing the UNIX operating system. Earlier UNIX was written in assembly language, but C made it easier to modify and transfer to different computers.**
- **C was derived from earlier programming languages like BCPL and B language.**
- **In 1989, ANSI standardized C, which became known as ANSI C. Later, improved versions like C99, C11, and C18 were introduced.**
- **Due to its speed, portability, and efficiency, C became one of the most widely used programming languages and is called the "Mother of Programming Languages."**
- **C is a structured programming language, which introduced the use of functions, loops, and control statements to make programs easier to write, understand, and maintain.**
- **C became the foundation for many modern programming languages such as C++, Java, and C# because of its powerful features and simple syntax.**

Q.Explain data types in C.

ANS:

Data types in C specify the type of data that a variable can store and determine the memory required for storing that data.

The data types in C are classified as:

1. **Data Types in C:**
2. **Data types specify the type of data that a variable can store and the amount of memory required.**
3. **Types of data types in C:**
4. **Basic Data Types:**
   Used to store simple values.
   Example: int, float, char, double
5. **Derived Data Types:**
   Created from basic data types.
   Example: Array, Pointer, Function
6. **User Defined Data Types:**
   Defined by the programmer.
   Example: Structure, Union, Enum
7. **Void Data Type:**
   Represents no value or empty data.

Explain tokens in C programming.

- **Tokens are the smallest individual units or basic building blocks of a C program that have a specific meaning to the compiler.**
- **A C program is made up of different types of tokens. They are classified as:**

1. **Keywords:**
   - **Reserved words that have a predefined meaning in C.**
   - **Examples: int, float, if, while, return**
2. **Identifiers:**
   - **Names given by the programmer to variables, functions, arrays, etc.**
   - **Examples: sum, marks, total**
3. **Constants:**
   - **Fixed values that cannot be changed during program execution.**
   - **Examples: 10, 3.14, 'A'**
4. **Strings:**
   - **A sequence of characters enclosed in double quotes.**
   - **Example: "Hello"**
5. **Operators:**
   - **Symbols used to perform operations on variables and values.**
   - **Examples: +, -, *, /, =**
6. **Special Symbols:**
   - **Symbols having special meaning in C programming.**
   - **Examples: { }, ( ), [ ], ;, #**

Q. List all the logical operators in C and provide an example of each.

Logical Operators in C:

- **Logical operators** are used to combine two or more conditions and perform logical operations.
- They return either **true (1)** or **false (0)** values.

Operator	Name	Example	Meaning
&&	Logical AND	(a > 5 && b < 10)	True if both conditions are true
`		`	Logical OR
!	Logical NOT	!(a > 5)	Reverses the result of condition

Example:

```
#include<stdio.h>

int main()
{
  int a = 10, b = 5;

  printf("%d", (a > 5 && b < 10));
  printf("%d", (a < 5 || b < 10));
  printf("%d", !(a > 5));

  return 0;
}
```

Output:

```
1
1
0
```

Q.What is the difference between a keyword and an identifier in C?

Keyword	Identifier
Keywords are predefined reserved words in C programming language.	Identifiers are user-defined names given to variables, functions, arrays, etc.
The meaning of keywords is already defined by the compiler.	The meaning of identifiers is defined by the programmer.
Keywords cannot be used as variable or function names.	Identifiers can be used as names of variables and functions.
C has a fixed number of keywords.	There can be unlimited identifiers created by users.
Keywords contain only lowercase letters.	Identifiers can contain letters, digits, and underscore (_).
Examples: int, float, if, while	Examples: sum, marks, total

Q. Explain the difference between a constant and a variable in C.

Constant	Variable
A constant is a fixed value that cannot be changed during program execution.	A variable is a named memory location whose value can be changed during program execution.
Constants must be initialized at the time of declaration.	Variables can be declared first and assigned values later.
The value of a constant remains the same throughout the program.	The value of a variable may change many times in a program.
Constants are declared using the const keyword or #define directive.	Variables are declared using data types like int, float, char, etc.
Example: const int a = 10;	Example: int a = 10; then a = 20;

Q.Difference between && (AND) and || (OR) Logical Operators in C.

&& (Logical AND)	|| (Logical OR)
Returns true only when both conditions are true.	Returns true when at least one condition is true.
If any condition is false, output becomes false.	If any condition is true, output becomes true.
Used when all conditions must be satisfied.	Used when any one condition must be satisfied.
It gives false when both conditions are false.	It gives false only when all conditions are false.
Symbol used is &&.	Symbol used is ||.
Example: (a>5 && b<10)	Example: (a>5 || b<10)

Q. List any three features that make C an important programming language.

Features of C Programming Language:

1. **Simple and Efficient:**
   - **C has a simple syntax and provides fast execution, making it an efficient programming language.**
2. **Portable Language:**
   - **C programs can run on different computer systems with little or no modification.**
3. **Structured Programming Language:**
   - **C supports functions and divides large programs into smaller modules, making programs easy to understand and maintain.**
4. **Rich Library Support:**
   - **C provides many built-in functions and libraries that help programmers develop applications easily.**
5. **Low-Level Memory Access:**
   - **C supports pointers, which allow direct memory access and make it suitable for system programming.**

Q.Define an identifier in C. Explain rules of naming an identifier.

dentifier in C:

- **An identifier is a user-defined name given to program elements such as variables, functions, arrays, structures, etc.**
- **It is used to uniquely identify these elements in a C program.**

Example: sum, total_marks, number1

Rules for Naming an Identifier:

1. **An identifier can contain letters (A-Z, a-z), digits (0-9), and underscore (_) only.**
2. **The first character of an identifier must be a letter or underscore, it cannot start with a digit.**
3. **Keywords cannot be used as identifiers because they are reserved words in C.**
4. **Identifiers are case-sensitive.**
   - **Example: Total and total are considered different identifiers.**
5. **Blank spaces and special symbols like @, #, %, & are not allowed.**
6. **Identifier names should be meaningful and unique.**

Example of valid identifiers: marks, student_name, _value

Example of invalid identifiers: 2num, float, total marks

Q. Define the following storage classes in C: auto, extern and static.

Storage Classes in C:

Storage classes in C define the scope, lifetime, visibility, and storage location of variables.

1. auto Storage Class:

- **auto is the default storage class for local variables.**
- **Variables declared inside a function are automatically considered as auto variables.**
- **Memory is allocated when the function starts and destroyed when the function ends.**

Example:

```
auto int a = 10;
```

2. extern Storage Class:

- **extern is used to declare a global variable that is defined in another file or another part of the program.**
- **It allows sharing of variables between multiple files.**
- **The lifetime of an extern variable is throughout the program execution.**

Example:

```
extern int count;
```

3. static Storage Class:

- **static variables retain their values even after the function execution is completed.**
- **Memory is allocated only once and remains available throughout the program.**
- **Default value of static variable is zero.**

Example:

```
static int x = 5;
```

QWhat is an arithmetic operator? Explain with example.

Arithmetic Operator in C:

- **Arithmetic operators are symbols used to perform mathematical calculations on operands (variables or values).**
- **They are used for operations such as addition, subtraction, multiplication, division, etc.**

Types of Arithmetic Operators:

Operator	Meaning	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus (Remainder)	a % b

Example:

```
#include<stdio.h>
int main()
{
  int a = 10, b = 5;

  printf("Addition = %d", a + b);
  printf("Subtraction = %d", a - b);

  return 0;
}
```

Output:

```
Addition = 15
Subtraction = 5
```

Q.Explain symbolic constant with example.

Symbolic Constant in C:

- **A symbolic constant is a name given to a fixed value that cannot be changed during program execution.**
- **It is defined using the #define preprocessor directive in C.**
- **Symbolic constants make programs easier to read, understand, and modify.**
- **They are usually written in capital letters for better identification.**

Syntax:

```
#define CONSTANT_NAME value
```

Example:

```
#include<stdio.h>
#define PI 3.14

int main()
{
  float area, r = 5;
  area = PI * r * r;
  printf("Area = %f", area);
  return 0;
}
```

Q. Explain any 4 bitwise operators in C.

Bitwise Operators in C:

- **Bitwise operators are used to perform operations on data at the binary (bit) level.**

Operator	Name	Description	Example
&	Bitwise AND	Gives 1 if both bits are 1	a & b
`	`	Bitwise OR	Gives 1 if any one bit is 1
^	Bitwise XOR	Gives 1 if both bits are different	a ^ b
~	Bitwise NOT	Converts 0 to 1 and 1 to 0	~a

Example:

```
int a = 5, b = 3;

a & b;
a | b;
a ^ b;
~a;
```

Q. Explain the difference between pre-increment (++i) and post-increment (i++) operators in C.

Pre-increment (++i)	Post-increment (i++)
Value is increased before using the variable.	Value is increased after using the variable.
First increment operation is performed, then value is assigned.	First value is assigned, then increment operation is performed.
Updated value is used in the expression.	Original value is used in the expression.
Syntax: ++i	Syntax: i++
Example: a = ++i;	Example: a = i++;
If i=5, then a=6, i=6	If i=5, then a=5, i=6

Q. Define an algorithm and write an algorithm for addition of two numbers.

Algorithm:

- An algorithm is a step-by-step procedure or set of instructions used to solve a particular problem.

Algorithm for Addition of Two Numbers:

Step 1: Start
Step 2: Declare variables a, b, and sum
Step 3: Read two numbers a and b
Step 4: Add the numbers: sum = a + b
Step 5: Display the value of sum
Step 6: Stop

Q.Define an operator and operand in C programming. Give a suitable example.

Operator:

- An operator is a symbol that performs a specific operation on one or more operands in a C program.
- Examples: +, -, *, /, %

Operand:

- An operand is a value or variable on which an operator performs an operation.
- Operands may be constants or variables.

Example:

```
int a = 10, b = 5, c;
c = a + b;
```

Here,

- + is an operator (performs addition).
- a and b are operands (values used in operation).

Q. Explain relational operators in C.

Relational Operators in C:

- Relational operators are used to compare two values or expressions.
- They return the result as true (1) or false (0).

Operator	Meaning	Example
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b
==	Equal to	a == b
!=	Not equal to	a != b

Q.What do you mean by operator precedence? Specify the precedence of arithmetic operators.

Operator Precedence:

- Operator precedence defines the order in which operators are evaluated in an expression.
- Operators with higher precedence are executed before operators with lower precedence.

Precedence of Arithmetic Operators:

Precedence	Operator	Operation
Highest	()	Parentheses
High	*, /, %	Multiplication, Division, Modulus
Low	+, -	Addition, Subtraction

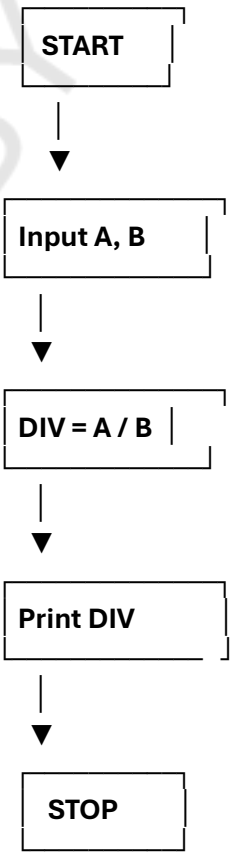
Example:

```
int x;
x = 5 + 3 * 2;
```

Here, * has higher precedence than +, so:

x = 5 + 6 = 11

Q.Draw a flowchart for division of two numbers.



Steps:

1. Start
2. Read two numbers A and B
3. Divide the numbers: DIV = A / B
4. Display the result
5. Stop

Q.Difference between if and if-else Statement in C.

if Statement	if-else Statement
Executes statement only when condition is true.	Executes one block when condition is true and another when false.
Contains only if block.	Contains both if and else blocks.
If condition is false, no output is produced.	If condition is false, else part is executed.
Used for single decision making.	Used for two-way decision making.
Syntax: if(condition)	Syntax: if(condition) else
Example: if(a>0) printf("Positive");	Example: if(a>0) printf("Positive"); else printf("Negative");

Q. Explain while loop with a suitable example.

While Loop in C:

- A while loop is an entry-controlled loop that executes a block of statements repeatedly until the given condition becomes false.
- The condition is checked before executing the loop body.
- It is used when the number of iterations is not fixed.

Syntax:

```
while(condition)
{
  // statements
}
```

Example:

```
#include<stdio.h>

int main()
{
  int i = 1;

  while(i <= 5)
  {
    printf("%d", i);
    i++;
  }
 return 0;
}
```

Output:

1 2 3 4 5

Operator	Operand
Operator is a symbol that performs a specific operation.	Operand is a value or variable on which the operator performs operation.
It specifies which operation should be performed.	It provides the data for performing the operation.
Operators cannot store values.	Operands can store values.
Examples of operators	Examples of operands are variables like a, b, x or values

Q. Explain simple if statement, if-else statement and nested if-else statement with suitable example.

1. Simple if Statement:

- It executes a block of code only when the given condition is true.

Example:

```
if(a > 0)
{
  printf("Positive number");
}
```

2. if-else Statement:

- It executes if block when condition is true, otherwise executes else block.

Example:

```
if(a % 2 == 0)
{
  printf("Even");
}
else
{
  printf("Odd");
}
```

3. Nested if-else Statement:

- An if-else statement inside another if-else statement is called nested if-else.

Example:

```
if(a > 0)
{
  printf("Positive");
}
else
{
  if(a < 0)
    printf("Negative");
  else
    printf("Zero");
}
```

Q. Write a program for finding factorial of a number.

```
#include<stdio.h>
int main()
{
  int n, i, fact = 1;
   printf("Enter a number: ");
  scanf("%d", &n);
   for(i = 1; i <= n; i++)
  {
    fact = fact * i;
  }

  printf("Factorial = %d", fact);

  return 0;
}
```

Output:

Enter a number: 5

Factorial = 120

Q. Explain for loop with continue and break statement.

For Loop:

- A for loop is used to execute a block of statements repeatedly until the given condition becomes false.
- It is mainly used when the number of iterations is known.

Syntax:

```
for(initialization; condition; increment/decrement)
{
  // statements
}
```

break Statement:

- The break statement is used to terminate the loop immediately and transfer control outside the loop.

Example:

```
for(i=1; i<=5; i++)
{
  if(i==3)
    break;

  printf("%d", i);
}
```

Output: 1 2

continue Statement:

- The continue statement skips the current iteration and continues with the next iteration of the loop.

Example:

```
for(i=1; i<=5; i++)
{
  if(i==3)
    continue;

  printf("%d", i);
}
```

Output: 1 2 4 5

Q. Write a program to check whether a number is even or odd.

```
#include<stdio.h>
int main()
{
  int n;
  printf("Enter a number: ");
  scanf("%d", &n);
   if(n % 2 == 0)
  {
    printf("Number is Even");
  }
  else
  {
    printf("Number is Odd");
  }
   return 0;
}
```

Output:

Enter a number: 6

Number is Even

Q.Describe how a "Switch" statement works and compare it to an "if-else" structure.

Switch Statement:

- A switch statement is a decision-making statement used to select one option from multiple choices.
- The expression value is compared with different case values.
- When a matching case is found, that block is executed.
- The break statement is used to exit from the switch block.

Difference between switch and if-else:

Switch Statement	if-else Statement
Used to select from multiple fixed choices.	Used for checking multiple conditions.
Works with equality comparison only.	Supports different conditions like <, >, <=, >=.
Uses case and break keywords.	Uses if and else keywords.
Easier to read for menu-based programs.	Suitable for complex conditions.

Q. Explain the syntax and usage of the goto statement in C with example.

goto Statement in C:

- The goto statement is a jump statement used to transfer the control of a program from one place to another.
- It transfers control to a specified label in the program.
- It is generally used to exit from loops or skip a part of the program.

Syntax:

```
goto label;

label:
  statement;
```

Example:

```
#include<stdio.h>

int main()
{
  int a = 5;

  if(a == 5)
    goto display;

  display:
    printf("Hello");

  return 0;
}
```

Output:

Hello

Q. Explain with an example the usage of shorthand assignment operators in C.

Shorthand Assignment Operators:

- Shorthand assignment operators are used to combine an arithmetic operation with an assignment operation.
- They reduce the length of expressions and make programs simpler.

Operator	Meaning	Example
+=	Add and assign	a += 5 → a = a + 5
-=	Subtract and assign	a -= 5 → a = a - 5
*=	Multiply and assign	a *= 5 → a = a * 5
/=	Divide and assign	a /= 5 → a = a / 5
%=	Modulus and assign	a %= 5 → a = a % 5

Example:

```
int a = 10;
a += 5;
printf("%d", a);
```

Output:

15

Q. Explain different conditional/branching statements in C programming.

Conditional/Branching Statements in C:

- Conditional statements are used to control the flow of execution depending on given conditions.

Types of Conditional Statements:

1. **if Statement:**
   - Executes a block of code only if the condition is true.

Example:

```
if(a > 0)
{
  printf("Positive");
}
```

2. **if-else Statement:**
   - Executes one block when condition is true and another block when condition is false.

Example:

```
if(a%2==0)
  printf("Even");
else
  printf("Odd");
```

3. **Nested if-else Statement:**
   - An if-else statement inside another if-else statement is called nested if-else.
4. **Switch Statement:**
   - Used to select one option from multiple choices using case statements.

Example:

```
switch(choice)
{
 case 1:
  printf("One");
  break;
}
```

Q. Explain one dimensional array with a suitable example.

One Dimensional Array in C:

- A one dimensional array is a collection of elements of the same data type stored in continuous memory locations.
- It stores multiple values using a single variable name.
- Each element is accessed using an index number, starting from 0.

Syntax:

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

Example:

```
#include<stdio.h>
int main()
{
  int a[5] = {10,20,30,40,50};
printf("%d", a[2]);

  return 0;
}
```

Output:

30

Q. Explain how an array is stored in memory and how elements can be accessed using an index.

Array Storage in Memory:

- An array stores elements of the same data type in continuous (contiguous) memory locations.
- Each array element has a unique index number used to access that element.
- The index of an array always starts from 0.

Example:

```
int a[5] = {10,20,30,40,50};
```

Memory Representation:

Index	Element	Value
a[0]	First element	10
a[1]	Second element	20
a[2]	Third element	30
a[3]	Fourth element	40
a[4]	Fifth element	50

Accessing Element:

```
printf("%d", a[2]);
```

Output:

30

Q. How do we define and call a function in C.

Function Definition:

- A function definition contains the actual code or statements that perform a specific task.
- It includes function name, return type, parameters, and function body.

Syntax:

```
return_type function_name(parameters)
{
  // statements
  return value;
```

Q. Write a program that asks the user to input 10 numbers, stores them in a one-dimensional array, and then prints the sum of the numbers.

```
#include<stdio.h>
int main()
{
   int a[10], i, sum = 0;
    printf("Enter 10 numbers:");
   for(i = 0; i < 10; i++)
   {
   scanf("%d", &a[i]);
   }
   for(i = 0; i < 10; i++)
   {
      sum = sum + a[i];
   }
  printf("Sum = %d", sum);
    return 0;
}
```

Output:

```
Enter 10 numbers:
1 2 3 4 5 6 7 8 9 10
Sum = 55
```

Q. Compare the strcpy() and strncpy() functions. Give suitable example.

strcpy()	strncpy()
It copies the entire string from source to destination.	It copies only a specified number of characters from source to destination.
It does not limit the number of characters copied.	It allows the user to specify the number of characters to copy.
Syntax: strcpy(destination, source);	Syntax: strncpy(destination, source, n);
Less safe because buffer overflow may occur.	Safer as it limits the copied characters.

Q. Write a program to concatenate two strings using the strcat() function.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[50], str2[50];
 printf("Enter first string: ");
   scanf("%s", str1);

   printf("Enter second string: ");
   scanf("%s", str2);

   strcat(str1, str2);

   printf("Concatenated string = %s", str1);

   return 0;
}
```

Q. Explain how to declare and initialize a two-dimensional array in C, with an example.

Two-Dimensional Array:

- **A two-dimensional array is a collection of elements arranged in rows and columns.**
- **It is also called a matrix.**

Declaration:

Syntax:

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

Example:

```
int a[2][3];
```

Initialization:

- **Values can be assigned at the time of declaration.**

Example:

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

Program Example:

```
#include<stdio.h>

int main()
{
   int a[2][2] = {{10,20},{30,40}};

   printf("%d", a[0][1]);

   return 0;
}
```

Output:

```
20
```

Q. Explain any five String handling Functions with example.

String Handling Functions in C:

1. **strlen()**

- It is used to find the length of a string.

Example:

strlen("Hello");

Output: 5

2. **strcpy()**

- It is used to copy one string into another string.

Example:

strcpy(str1, "Hello");

Output: str1 = Hello

3. **strcat()**

- It is used to join (concatenate) two strings.

Example:

strcat(str1, str2);

If str1 = "Hello" and str2 = "World"

Output: HelloWorld

4. **strcmp()**

- It is used to compare two strings.

Example:

strcmp("abc","abc");

Output: 0 (Strings are equal)

5. **strrev()**

- It is used to reverse a given string.

Example:

strrev("Hello");

Q. Write a program that reverses a string entered by the user without using built-in functions.

```
#include<stdio.h>
int main()
{
   char str[50], rev[50];
   int i, j, len = 0;
    printf("Enter a string: ");
   scanf("%s", str);
 while(str[len] != '\0')
   {
      len++;
   }
 j = 0;
   for(i = len-1; i >= 0; i--)
   {
      rev[j] = str[i];
      j++;
   }
 rev[j] = '\0';
printf("Reverse string = %s", rev);
 return 0;
}
```

Output:

```
Enter a string: Hello
Reverse string = olleH
```

Q. What are the different ways of string initialization?

String Initialization in C:

- String initialization means assigning values to a character array during declaration.

Different ways of string initialization:

1. Using character array:

```
char str[6] = {'H','e','l','l','o','\0'};
```

2. Using string literal:

```
char str[] = "Hello";
```

3. Specifying size with string:

```
char str[10] = "Hello";
```

4. Using pointer:

```
char *str = "Hello";
```

Q. Explain how a string in C is terminated and how this affects string operations.

String Termination in C:

- In C, every string is terminated by a special character called null character (\0).
- The null character indicates the end of the string.
- It helps string functions to identify where the string stops.

Example:

```
char str[] = "Hello";
```

Memory representation:

H	e	l	l	o	\0

Effect on String Operations:

- Functions like strlen(), strcpy(), strcmp() read characters until they find \0.
- If \0 is missing, string functions may access extra memory and give incorrect results.

Q. Write a program in 'C' for addition of two matrices.

```
#include<stdio.h>
int main()
{
   int a[2][2], b[2][2], c[2][2];
   int i, j;
 printf("Enter first matrix:");
   for(i=0; i<2; i++)
   {
      for(j=0; j<2; j++)
      {
         scanf("%d", &a[i][j]);
      }
   }
 printf("Enter second matrix:");
   for(i=0; i<2; i++)
   {
      for(j=0; j<2; j++)
      {
         scanf("%d", &b[i][j]);
      }
   }
  for(i=0; i<2; i++)
   {
      for(j=0; j<2; j++)
      {
         c[i][j] = a[i][j] + b[i][j];
      }
   }
printf("Addition of Matrix:\n");
   for(i=0; i<2; i++)
   {
      for(j=0; j<2; j++)
      {
         printf("%d ", c[i][j]);
      }
      printf("\n");
   }
 return 0;
}
```

Output:

```
First Matrix:
1 2
3 4

Second Matrix:
5 6
7 8

Addition:
6 8
10 12
```

Q. Write a C Program to Check if a Given String is Palindrome.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[50];
  int i, len, flag = 0;
 printf("Enter a string: ");
  scanf("%s", str);
 len = strlen(str);
 for(i = 0; i < len/2; i++)
  {
    if(str[i] != str[len-i-1])
    {
      flag = 1;
      break;
    }
  }
 if(flag == 0)
    printf("String is Palindrome");
  else
    printf("String is Not Palindrome");
return 0;
}
```

Output:

```
Enter a string: madam
String is Palindrome
```

Q. Explain the string functions - strcpy, strlen, strcmp, strncmp, strcat.

String Functions in C:

1. strcpy():

- It is used to copy one string into another string.

Example:

```
strcpy(str1, "Hello");
```

Output: str1 = Hello

2. strlen():

- It is used to find the length of a string.

Example:

```
strlen("Hello");
```

Output: 5

3. strcmp():

- It is used to compare two strings.
- It returns 0 if both strings are equal.

Example:

```
strcmp("abc", "abc");
```

Output: 0

4. strncmp():

- It compares only the specified number of characters of two strings.

Example:

```
strncmp("Hello", "Help", 3);
```

Output: 0 (First 3 characters are same)

5. strcat():

- It is used to join two strings together.

Example:

```
strcat("Hello", "World");
```

Output: HelloWorld

Q. Explain built-in functions and user-defined functions in C programming.

1. Built-in Functions:

- The functions which are already defined in C libraries are called built-in functions.
- They can be directly used by including header files.

Examples:

```
printf();   // displays output
scanf();    // takes input
strlen();   // finds string length
```

2. User-defined Functions:

- The functions which are created by the programmer according to the requirement are called user-defined functions.
- They help to divide a large program into smaller parts.

Example:

```
int add(int a, int b)
{
  return a+b;
}
```

Q. Write a program to declare a structure for an employee with members for name, age and gender. Initialize the structure and print the values.

```
#include<stdio.h>
struct employee
{
  char name[20];
  int age;
  char gender;
};

int main()
{
  struct employee e1 = {"Pratik", 21, 'M'};

  printf("Employee Name = %s\n", e1.name);
  printf("Employee Age = %d\n", e1.age);
  printf("Employee Gender = %c", e1.gender);

  return 0;
}
```

Output:

```
Employee Name = Pratik
Employee Age = 21
Employee Gender = M
```

Q. Explain user defined function definition with syntax and give a suitable example.

User Defined Function:

- **A user defined function is a function created by the programmer to perform a specific task.**
- **It helps to divide a large program into smaller modules and improves code reusability.**

Syntax:

```
return_type function_name(parameters)
{
  // function body
  statements;
return value;
}
```

Example:

```
#include<stdio.h>
int add(int a, int b)
{
  int sum;
  sum = a + b;
  return sum;
}
int main()
{
  printf("%d", add(5, 3));
  return 0;
}
```

Output:

```
8
```

Q. Write a program for leap year using function.

```
#include<stdio.h>

void leap(int year)
{
  if((year%4==0 && year%100!=0) || year%400==0)
    printf("Leap Year");
  else
    printf("Not Leap Year");
}

int main()
{
  int year;

  printf("Enter year: ");
  scanf("%d",&year);

  leap(year);

  return 0;
}
```

Output:

```
Enter year: 2024
Leap Year
```

Q. What is recursion? Explain it with a suitable example.

Recursion:

- **Recursion is a process in which a function calls itself repeatedly until a specific condition is satisfied.**
- **A function that calls itself is called a recursive function.**
- **It reduces code length and helps solve complex problems easily.**

Example:

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

Here, the function factorial() calls itself, so it is called recursion.

Q.Explain structure declaration and initialization.

Structure Declaration:

- **A structure is a user-defined data type in C that stores different types of data under a single name.**
- **It is declared using the struct keyword.**

Syntax:

```
struct structure_name
{
  data_type member1;
  data_type member2;
};
```

Structure Initialization:

- **Assigning values to structure members is called structure initialization.**
- **Example:**

```
struct student
{
  int roll;
  char name[20];
};
struct student s1 = {1, "Pratik"};
```

Array	String
Array is a collection of elements of the same data type stored in continuous memory locations.	String is a collection of characters stored in continuous memory locations.
Array can store different types of data like int, float, char, etc.	String stores only characters (char data type).

Q.Explain function declaration, function definition and function call. Give a suitable example.

1. Function Declaration:

- **It tells the compiler about the function name, return type and parameters.**

Example:

```
int add(int, int);
```

2. Function Definition:

- **It contains the actual statements executed by the function.**

Example:

```
int add(int a, int b)
{
  return a+b;
}
```

3. Function Call:

- **It is used to execute the defined function.**

Example:

```
sum = add(5,10);
```

Complete Example:

```
#include<stdio.h>
int add(int,int);
int add(int a,int b)
{
  return a+b;
}
int main()
{
  printf("%d", add(5,10));
  return 0;
}
```

Output: 15

Q.Write a program for prime number using function.

```
#include<stdio.h>
void prime(int n)
{
  int i, flag=0;

  for(i=2; i<n; i++)
  {
    if(n%i==0)
    {
      flag=1;
      break;
    }
  }
 if(flag==0)
    printf("Prime Number");
  else
    printf("Not Prime Number");
}
int main()
{
  int n;
 printf("Enter number:");
  scanf("%d",&n);
 prime(n);
 return 0;
}
```

Q. Write a C program that accepts two numbers from the user and swaps them using a function.

```
#include<stdio.h>

void swap(int a, int b)
{
  int temp;
 temp = a;
  a = b;
  b = temp;

  printf("After swapping: a=%d b=%d", a, b);
}
int main()
{
  int a, b;
  printf("Enter two numbers: ");
  scanf("%d%d", &a, &b);
  printf("Before swapping: a=%d b=%d\n", a, b);
  swap(a, b);
 return 0;
}
```

Q. Explain the concept of return values in C. How is the return type of a function determined? Provide an example of a function that returns an integer value.

Return Values in C:

- **A return value is the value sent back by a function to the calling function.**
- **The return statement is used to return a value from a function.**
- **The data type of the returned value is specified by the function return type.**
- **The return type is determined according to the type of value returned, such as int, float, char, or void.**

Example (Integer Return Value):

```
#include<stdio.h>

int add(int a, int b)
{
  int sum;
  sum = a + b;
  return sum;
}

int main()
{
  int result;

  result = add(5,10);

  printf("Sum = %d", result);

  return 0;
}
```

Output:

Sum = 15