

SUMMER 2023 EXAMINATION MODEL ANSWER

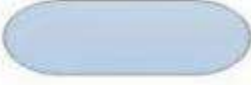




Subject: Programming in 'C'

Subject Code: 22226

1. Attempt any FIVE of the following: 10 M

a) List different symbols used in flowchart(any 4)

ANS:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

b) Give the use of printf()

ANS:

printf() & scanf():

printf() and scanf() functions are library functions in C programming language defined in "stdio.h" header file.

Uses of printf():

1. It is used to print the any message onto the output screen.
2. It is used to print formatted text and values to the standard output stream.

c) Give the syntax of for loop

ANS:

The syntax of for loop in c language is given below:

```
for(Expression 1; Expression 2; Expression 3)
{ //code to be executed. }
```

d) Define Array

ANS:

An array is a collection of data items, all of the same type, accessed using a common name.

- 1) A one-dimensional array consists of similar type of multiple values in it.
- 2) A two dimensional array consists of row and column.

e) List any two string handling function

ANS: `strlen()`: calculates the length of the string.

Syntax: `strlen(s1)`;

`strcat()`: concatenates two strings

Syntax: `strcat(s1,s2)`

f) Explain pointer with example.

ANS: **Definition:**

A pointer is a variable that stores memory address of another variable which is of similar data type.

Declaration:

`datatype *pointer_variable_name;`

Eg: `int *ptr;`

g) Define algorithm.

ANS: Algorithm is a stepwise set of instructions written to perform a specific task..

Q-2) Attempt any Three of the following: 12 M

a) Write a C program to find greatest number among three numbers.

ANS: **Program:**

```
#include <stdio.h>
int main()
{
double n1, n2, n3;
printf("Enter three numbers: ");
scanf("%lf %lf %lf", &n1, &n2, &n3);
// if n1 is greater than both n2 and n3, n1 is the largest
if (n1 >= n2 && n1 >= n3)
printf("%.2lf is the largest number.", n1);
// if n2 is greater than both n1 and n3, n2 is the largest
else if (n2 >= n1 && n2 >= n3)
printf("%.2lf is the largest number.", n2);
// if both above conditions are false, n3 is the largest
else
printf("%.2lf is the largest number.", n3);
return 0;
}
```

OUTPUT

Enter three numbers: 10 11 15
15.00 is the largest number.

b) Explain Go-to statement with example.

ANS: The C goto statement is a jump statement which is sometimes also referred to as an unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a function.

Syntax:

Syntax1

```
goto label;
```

```
    |
```

```
·
```

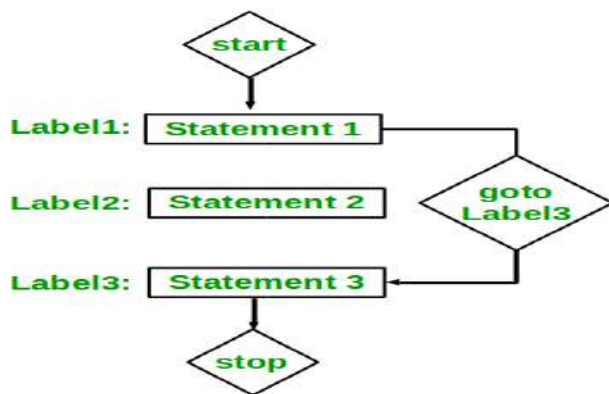
```
·
```

```
·
```

```
label:
```

In the above syntax, the first line tells the compiler to go to or jump to the statement marked as a label. Here, the label is a user-defined identifier that indicates the target statement.

The statement immediately followed after 'label:' is the destination statement. The 'label:' can also appear before the 'goto label;' statement in the above syntax.



Flowchart of goto Statement in C

Examples:

Type 1: In this case, we will see a situation similar to as shown in Syntax1 above. Suppose we need to write a program where we need to check if a number is even or not and print accordingly using the goto statement. The below program explains how to do this:

```
// from 1 to 10 using goto statement
#include <stdio.h>
// function to print numbers from 1 to 10
void printNumbers()
{
int n = 1;
label:
printf("%d ", n);
n++;
if (n <= 10)
goto label;
}
// Driver program to test above function
int main()
{
printNumbers();
return 0;
}
```

Output:

1 2 3 4 5 6 7 8 9 10

c) Write 'C' program to add two distances given in km using structure.

```
ANS: #include <stdio.h>
struct Distance
{
int feet;
float inch;
} firstDistance, secondDistance, sum;
int main()
```

```

{
printf("Enter feet and inches for the first distance: \n");
scanf("%d %f", &firstDistance.feet, &firstDistance.inch);
printf("Enter feet and inches for the second distance: \n");
scanf("%d %f", &secondDistance.feet, &secondDistance.inch);
sum.feet = firstDistance.feet + secondDistance.feet;
sum.inch = firstDistance.inch + secondDistance.inch;
while (sum.inch >= 12)
{
sum.inch = sum.inch - 12;
sum.feet++;
}
printf("The Sum is %d feet, %.1f inches\n", sum.feet, sum.inch);
return 0;
}

```

Output:

```

Enter feet and inches for the first distance:
12.4
Enter feet and inches for the second distance:
4.1
The Sum is 16 feet, 0.5 inches

...Program finished with exit code 0
Press ENTER to exit console.

```

d) Write a program to calculate sum of all elements stored in given array using pointers.

ANS: #include <stdio.h>

#include <malloc.h>

void main()

{

```
int i, n, sum = 0;
int *a;
printf("Enter the size of array A \n");
scanf("%d", &n);
a = (int *) malloc(n * sizeof(int));
printf("Enter Elements of the List \n");
for (i = 0; i < n; i++)
{
scanf("%d", a + i);
}
/* Compute the sum of all elements in the given array */
for (i = 0; i < n; i++)
{
sum = sum + *(a + i);
/* this *(a+i) is used to access the value stored at          the address*/
}
printf("Sum of all elements in array = %d\n", sum);
return 0;
}
```

Enter the size of array A

5

Enter Elements of the List

4

9

10

56

100

Sum of all elements in array = 179

Q-3) Attempt any Three of the following: 12 M

a) Explain use of comment in C language.

ANS:

Comments in C

Comments can be used to explain code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Comments can be **singled-lined** or **multi-lined**.

Single-line Comments

Single-line comments start with two forward slashes (`//`).

Any text between `//` and the end of the line is ignored by the compiler (will not be executed).

This example uses a single-line comment before a line of code:

Example

```
// This is a comment  
printf("Hello World!");
```

This example uses a single-line comment at the end of a line of code:

Example

```
printf("Hello World!"); // This is a comment
```

C Multi-line Comments

Multi-line comments start with `/*` and ends with `*/`.

Any text between `/*` and `*/` will be ignored by the compiler:

Example

```
/* The code below will print the words Hello World!  
to the screen, and it is amazing */  
printf("Hello World!");
```

b) Explain any two math function with syntax and give example of each.

ANS:

The C Math Functions can be classified into two main categories: Basic Math Functions and Advanced Math Functions. In addition to these main categories, there are also a number of utility functions that are used for specific applications. Let's delve deeper into these categories.

Basic Math Functions in C

Basic Math Functions in C are those that perform elementary mathematical operations, such as addition, subtraction, multiplication, division, and modulus. Some of these Basic Math Functions include:

- `abs()` - returns the absolute value of an integer
- `ceil()` - rounds up a floating-point number to the nearest integer
- `floor()` - rounds down a floating-point number to the nearest integer
- `pow()` - raises a number to the power of another number
- `sqrt()` - calculates the square root of a number

These Basic Math Functions can handle most simple arithmetic operations in your programs.

Advanced Math Functions in C

Advanced Math Functions in C are more complex and cater to specific mathematical needs, such as trigonometric, logarithmic, and exponential operations. Some examples of Advanced Math Functions include:

- `sin()`, `cos()`, `tan()` - calculate trigonometric sine, cosine, and tangent
- `asin()`, `acos()`, `atan()` - calculate inverse trigonometric sine, cosine, and tangent
- `exp()` - calculates the exponential value of a number
- `log()`, `log10()` - calculate natural and base-10 logarithms
- `hypot()` - calculates the square root of the sum of the squares of two numbers (useful in calculating the hypotenuse of a right-angled triangle)

These Advanced Math Functions are typically used in more specialized tasks and scientific applications where a higher level of precision is required.

c) Describe use of header files in C language.

ANS:

In **C language**, header files contain a set of predefined standard library functions. The **.h** is the extension of the header files in C and we request to use a header file in our program by including it with the C preprocessing directive “**#include**”.

C Header files offer the features like library functions, data types, macros, etc by importing them into the program with the help of a preprocessor directive “**#include**”.

Syntax of Header Files in C

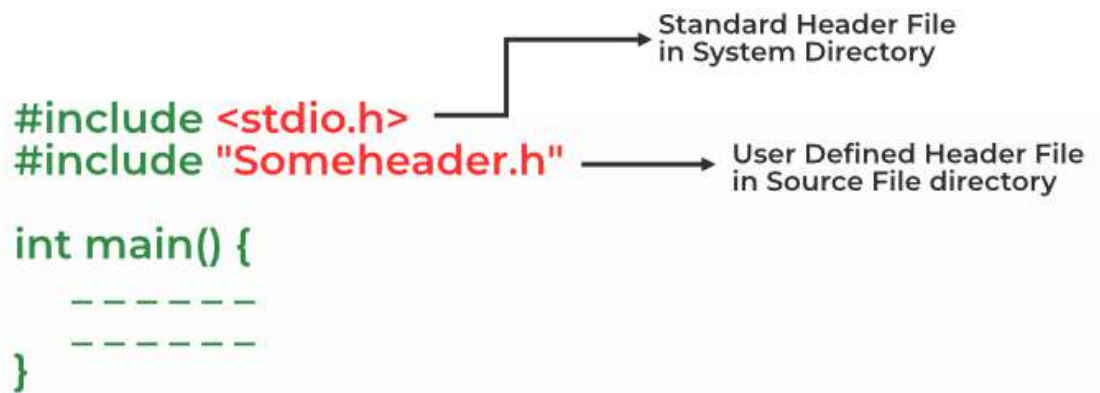
We can include header files in C by using one of the given two syntax whether it is a pre-defined or user-defined header file.

```
#include <filename.h> // for files in system/default directory
```

or

```
#include "filename.h" // for files in same directory as source file
```


The “#include” preprocessor directs the compiler that the header file needs to be processed before compilation and includes all the necessary data types and function definitions.



```
// C program to demonstrate the use of header files  
// standard input and output stdio.h header file  
#include <stdio.h>
```

```
int main()  
{  
printf(  
"Printf() is the function in stdio.h header file");  
return 0;  
}
```

d) Explain call by value with an example.

ANS:

The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

By default, C programming uses *call by value* to pass arguments. In general, it means the code within a function cannot alter the arguments used to call the function. Consider the function **swap()** definition as follows.

```
/* function definition to swap the values */  
void swap(int x, int y) {  
  
int temp;  
  
temp = x; /* save the value of x */  
x = y; /* put y into x */  
y = temp; /* put temp into y */  
  
return;  
}
```

Now, let us call the function **swap()** by passing actual values as in the following example –

[Live Demo](#)

```
#include <stdio.h>

/* function declaration */
void swap(int x, int y);

int main () {

/* local variable definition */
int a = 100;
int b = 200;

printf("Before swap, value of a : %d\n", a );
printf("Before swap, value of b : %d\n", b );

/* calling a function to swap the values */
swap(a, b);

printf("After swap, value of a : %d\n", a );
printf("After swap, value of b : %d\n", b );

return 0;
}
void swap(int x, int y) {

int temp;

temp = x; /* save the value of x */
x = y; /* put y into x */
y = temp; /* put temp into y */

return;
}
```

Let us put the above code in a single C file, compile and execute it, it will produce the following result –

```
Before swap, value of a : 100
Before swap, value of b : 200
After swap, value of a : 100
After swap, value of b : 200
```

Q-4) Attempt any Three of the following: 12 M

a) Write a C program to determine whether a given number is prime or not.

ANS:

```
#include <stdio.h> // Include the standard input/output header file.

int main() {

int num, i, ctr = 0; // Declare variables for user input, loop control, and a counter.

// Prompt the user to input a number.
printf("Input a number: ");
scanf("%d", &num); // Read the input number from the user.

// Start a loop to check for factors of the input number.
for (i = 2; i <= num / 2; i++) {
if (num % i == 0) { // If the input number is divisible by 'i'.
ctr++; // Increment the counter.
break; // Exit the loop since a factor has been found.
}
}

// If no factors were found and the number is not 1.
if (ctr == 0 && num != 1)
printf("%d is a prime number.\n", num); // Print that the number is prime.
else
printf("%d is not a prime number.\n", num); // Print that the number is not prime.

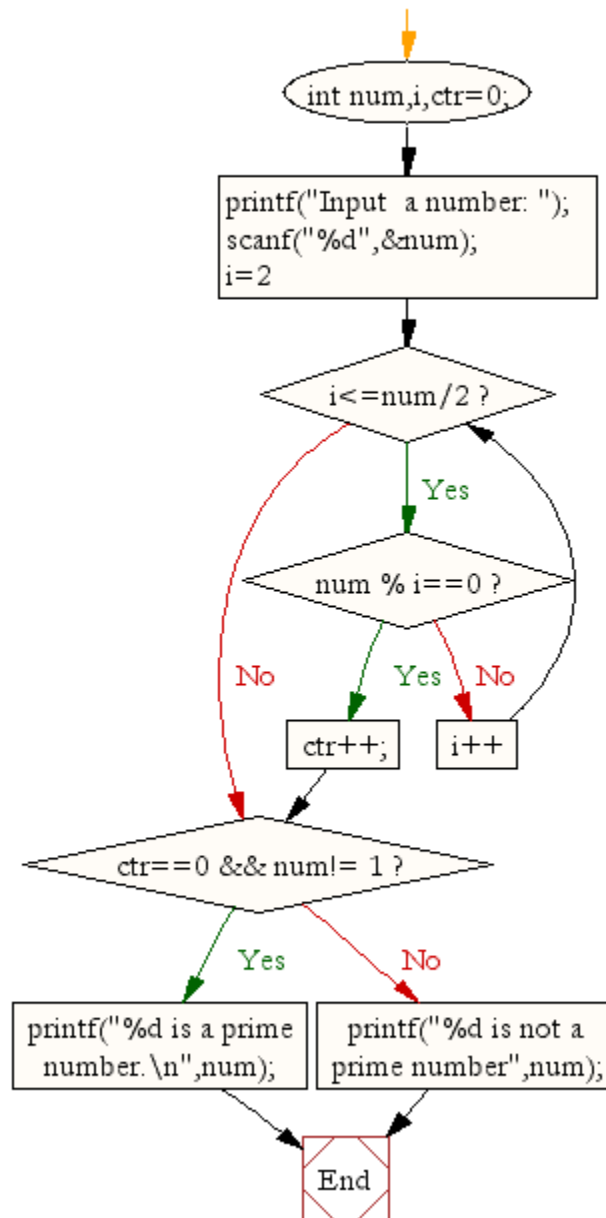
return 0; // Return 0 to indicate successful execution.
}
```

Copy

Sample Output:

Input a number: 13
13 is a prime number.

Flowchart:



b) Explain formatted input-output function with example in C.

ANS: This article focuses on discussing the following topics in detail-

- Formatted I/O Functions.
- Unformatted I/O Functions.
- Formatted I/O Functions vs Unformatted I/O Functions.

Formatted I/O Functions

[Formatted I/O functions](#) are used to take various inputs from the user and display multiple outputs to the user. These types of I/O functions can help to display the output to the user in different formats using the format specifiers. These I/O supports all [data types](#) like int, float, char, and many more.

Why they are called formatted I/O?

These functions are called formatted I/O functions because we can use format specifiers in these functions and hence, we can format these functions according to our needs.

List of some format specifiers-

S NO.	Format Specifier	Type	Description
1	%d	int/signed int	used for I/O signed integer value
2	%c	char	Used for I/O character value
3	%f	float	Used for I/O decimal floating-point value
4	%s	string	Used for I/O string/group of characters
5	%ld	long int	Used for I/O long signed integer value
6	%u	unsigned int	Used for I/O unsigned integer value
7	%i	unsigned int	used for the I/O integer value
8	%lf	double	Used for I/O fractional or floating data
9	%n	prints	prints nothing

c) Explain do-while loop in C with proper example.

ANS:

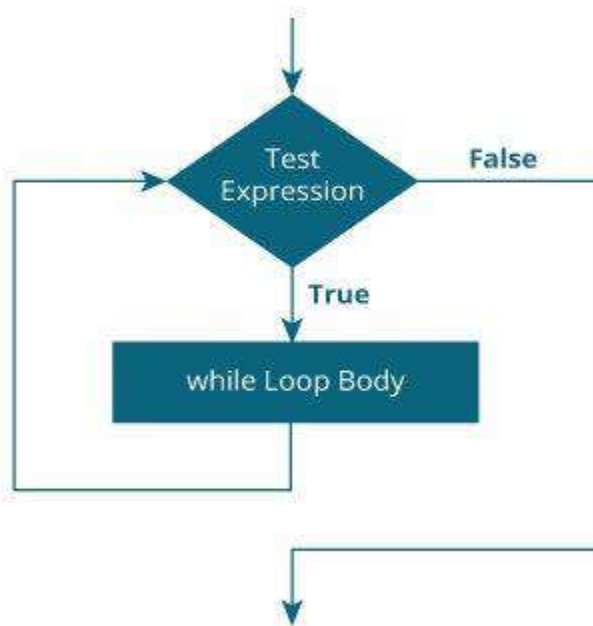
while loop

The syntax of the `while` loop is:

```
while (testExpression) {  
  // the body of the loop  
}
```

How while loop works?

- The `while` loop evaluates the `testExpression` inside the parentheses `()`.
- If `testExpression` is **true**, statements inside the body of `while` loop are executed. Then, `testExpression` is evaluated again.
- The process goes on until `testExpression` is evaluated to **false**.
- If `testExpression` is **false**, the loop terminates (ends).



d) Write a C program for multiplication of two 3*3 matrices.

ANS:

```
#include<stdio.h>
```

```
int main()
{
int i,j,k;
float a[3][3], b[3][3], mul[3][3];

printf("Enter elements of first matrix:\n");
for(i=0;i< 3;i++)
{
for(j=0;j< 3;j++)
{
printf("a[%d][%d]=",i,j);
scanf("%f", &a[i][j]);
}
}

printf("Enter elements of second matrix:\n");
for(i=0;i< 3;i++)
{
for(j=0;j< 3;j++)
{
printf("b[%d][%d]=",i,j);
scanf("%f", &b[i][j]);
}
}

for(i=0;i< 3;i++)
{
for(j=0;j< 3;j++)
{
mul[i][j] = 0;
for(k=0;k< 3;k++)
{
mul[i][j] = mul[i][j] + a[i][k]*b[k][j];
}
}
}
}
```

```

printf("Multiplied matrix is:\n");
for(i=0;i< 3;i++)
{
for(j=0;j< 3;j++)
{
printf("%f\t", mul[i][j]);
}
printf("\n");
}
return 0;
}

```

e) Write a C program to generate Fibonacci series for a given number using recursion.

ANS:

1. #include<stdio.h>
2. int main()
3. {
4. int n1=0,n2=1,n3,i,number;
5. printf("Enter the number of elements:");
6. scanf("%d",&number);
7. printf("\n%d %d",n1,n2);//printing 0 and 1
8. for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
9. {
10. n3=n1+n2;
11. printf(" %d",n3);
12. n1=n2;
13. n2=n3;
14. }
15. return 0;
16. }

Q-5) Attempt any Two of the following: 12 M

a) Write a C program to print following pattern

```

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```



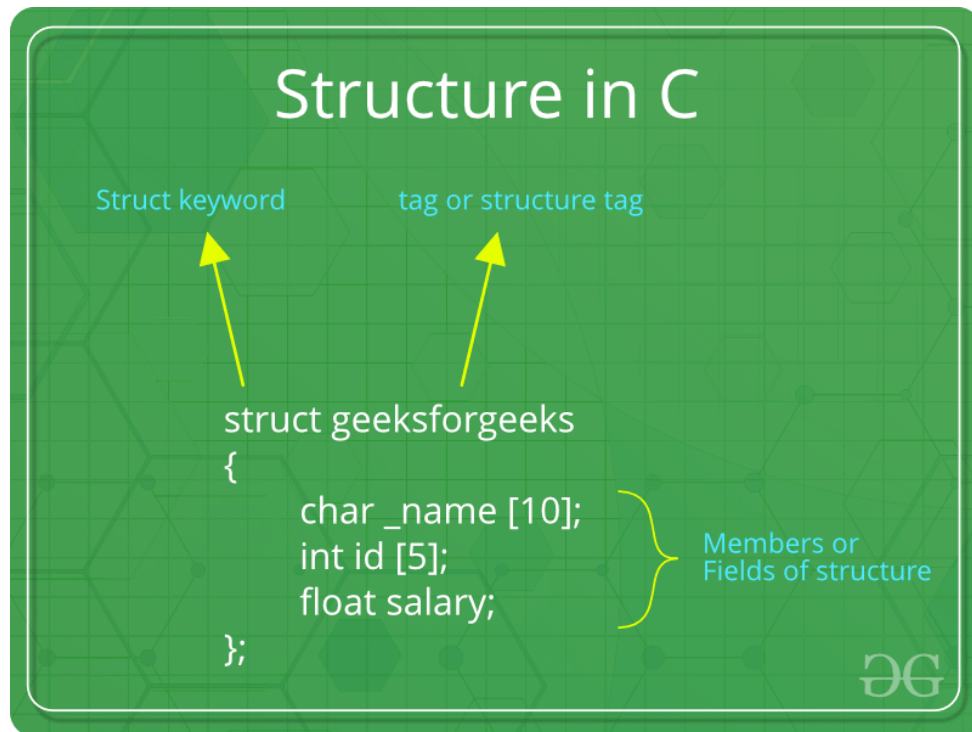
```

ANS: include <stdio.h>
int main() {
int i, j, rows;
printf("Enter the number of rows: ");
scanf("%d", &rows);
for (i = 1; i <= rows; ++i) {
for (j = 1; j <= i; ++j) {
printf("* ");
}
printf("\n");
}
return 0;
}

```

b) Explain how to initialize and define structure in C programming.

ANS:



C Structure Declaration

We have to declare structure in C before using it in our program. In structure declaration, we specify its member variables along with their datatype. We can use the struct keyword to declare the structure in C using the following syntax:

Syntax

```

struct structure_name {
    data_type member_name1;
    data_type member_name1;
    ....
    ....
};

```

The above syntax is also called a structure template or structure prototype and no memory is allocated to the structure in the declaration.

C Structure Definition

To use structure in our program, we have to define its instance. We can do that by creating variables of the structure type. We can define structure variables using two methods:

Structure Variable Declaration with Structure Template

```

struct structure_name {
    data_type member_name1;
    data_type member_name1;
    ....
    ....
}variable1, variable2, ...;

```

c)Develop a C program to find sum of all elements stored in given array using pointers.

ANS:

```

1. #include <stdio.h>
2. #include <malloc.h>
3.
4. void main()
5. {
6. int i, n, sum = 0;
7. int *a;
8.
9. printf("Enter the size of array A \n");
10. scanf("%d", &n);
11.
12. a = (int *) malloc(n * sizeof(int));
13.
14. printf("Enter Elements of the List \n");
15. for (i = 0; i < n; i++)
16. {
17. scanf("%d", a + i);
18. }
19.
20. /* Compute the sum of all elements in the given array */
21.
22. for (i = 0; i < n; i++)
23. {
24. sum = sum + *(a + i);
25. /* this *(a+i) is used to access the value stored at the address*/
26. }

```

```
27. printf("Sum of all elements in array = %d\n", sum);
28. return 0;
29. }
```

Q-6) Attempt any Two of the following: 12 M

a) Write a C program to find the largest and smallest number in a given array.

ANS:

```
1. #include <stdio.h>
2.
3. int main() {
4. // Declare an array
5. int arr[] = {12, 45, 3, 67, 89, 34, 56, 23};
6.
7. // Determine the size of the array
8. int size = sizeof(arr) / sizeof(arr[0]);
9.
10. // Initialize variables to store the smallest and largest numbers
11. int smallest = arr[0];
12. int largest = arr[0];
13.
14. // Iterate through the array to find the smallest and largest numbers
15. for (int i = 1; i < size; i++) {
16. if (arr[i] < smallest) {
17. smallest = arr[i];
18. }
19. if (arr[i] > largest) {
20. largest = arr[i];
21. }
22. }
23.
24. // Print the smallest and largest numbers
25. printf("Smallest number: %d\n", smallest);
26. printf("Largest number: %d\n", largest);
27.
28. return 0;
29. }
```

b) Write a C program to find factorial of a number using recursion.

ANS:

```
1. #include<stdio.h>
2. int main()
3. {
4. int i,fact=1,number;
```

```
5. printf("Enter a number: ");
6. scanf("%d",&number);
7. for(i=1;i<=number;i++){
8. fact=fact*i;
9. }
10. printf("Factorial of %d is: %d",number,fact);
11. return 0;
12. }
```

Output:

```
Enter a number: 5
Factorial of 5 is: 120
```

c) Write a C program to demonstrate access structure members using pointer.

ANS:

```
// C program to demonstrate structure pointer
```

```
#include <stdio.h>
```

```
struct point {
int value;
};
```

```
int main()
{
```

```
struct point s;
```

```
// Initialization of the structure pointer
```

```
struct point* ptr = &s;
```

```
return 0;
}
```