

Delhi Technological University

Department Of Computer Science And Engineering



Programming Fundamentals

Lab File for the course CO102 (P)

Submitted by : Vishal Das (2K21/A8/24)

Submitted to : Anukriti Kaushal

Contents

S. No.	Program	Date	Signature
1	Write a C program to print "Hello world" on screen	May 05, 2022	
2	Write a C program to take input integer from user and add one to that integer and print result		
3	Write a C program to add, subtract, multiplication and division on two integers and print the result and use explicit type casting for division operation		
4	Write a C program to calculate simple interest and print on the screen		
5	Write a C program to perform Logical AND and OR operations on two and three variables	May 12, 2022	
6	Write a C program to perform bitwise AND and OR operations on two variables		
7	Write a C program to perform relational operations on two variables		
8	Write a C program to compute factorial of a number	May 19, 2022	
9	Write a C program to compute sum of a 5-digit number		
10	Write a C program to reverse a 5-digit number		
11	Write a C program to find whether the input number is prime or not		
12	Write a program to find area of different shapes using switch case	June 09, 2022	
13	Write a program to implement following pattern :		

	<ul style="list-style-type: none"> • Half Pyramid of * • Half Pyramid of numbers • Inverted half pyramid of * • Inverted half pyramid of numbers • Full Pyramid of * 		
14	Write a program to convert decimal to binary and vice versa	June 16, 2022	
15	Program to generate the Fibonacci sequence	June 23, 2022	
16	Program to search a number from an array using linear search		
17	Program to search a number from an array using binary search		
18	Program to sort an array using Bubble sort		
19	Program to sort an array using Insertion sort		
20	Program to sort an array using Selection sort		
21	Program to check if a given string is a palindrome or not	July 07, 2022	
22	Program to string concatenation		
23	Program to string comparison		
24	Program to string reverse		
25	Program to convert a string from lower case to upper case and vice versa		
26	Program to find factorial of a number using recursion	July 14, 2022	
27	Write a C program for the addition of two 3 x 3 matrices		
28	Write a C program to multiply two 3 x		

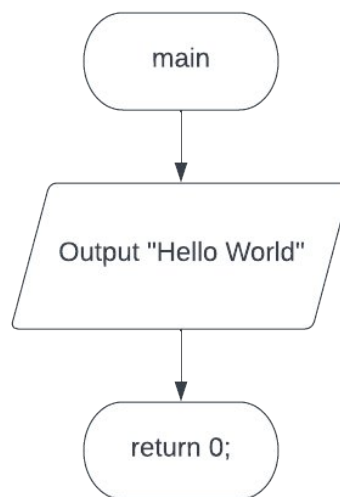
	3 matrices		
29	Write a C program to store the employee details using structure	July 14, 2022	
30	Write a C program to swap two numbers using pointers	July 21, 2022	
31	Write a C program to pass and return pointer to function hence calculate average of an array.		
32	Write a C program to create a file called <i>emp.txt</i> and store information about a person, in terms of his name, age and salary		
33	Write a C program to read a file and after converting all lower case to upper case letters write it to another file		

Program 01 : Write a C program to print "Hello world" on screen

Theory :

printf() function is used to write the C string pointed by format to the standard output.

Algorithm :



Code :

```
/* This program is written by Vishal Das  
(2K21/A8/24) on May 05, 2022 02:00 PM  
for Lab File of course C0102  
*/
```

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

```
int main(void){  
    printf("Hello World!");  
    return 0;  
}
```

Output :

```
Hello World!  
Process returned 0 (0x0)   execution time : 0.005 s  
Press ENTER to continue.  
█
```

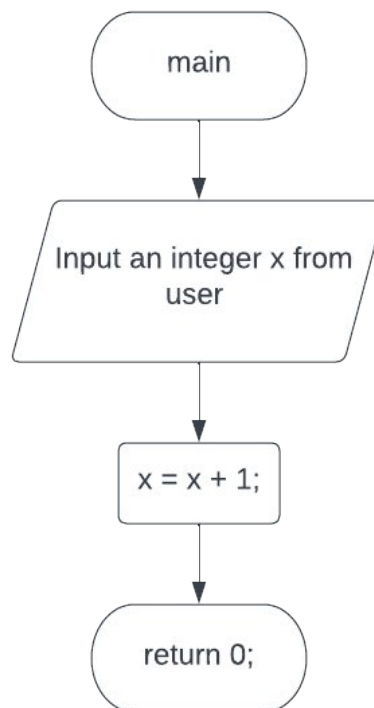
The screenshot shows a terminal window with a white background and a grey border. It contains the output of the program: 'Hello World!'. Below this, it shows 'Process returned 0 (0x0)' and 'execution time : 0.005 s'. At the bottom, it says 'Press ENTER to continue.' followed by a black cursor line.

Program 02 : Write a C program to take input integer from user and add one to that integer and print result.

Theory :

scanf() function is used to read data from user input and stores them according to the parameter format into the locations pointed by the additional arguments.

Algorithm :

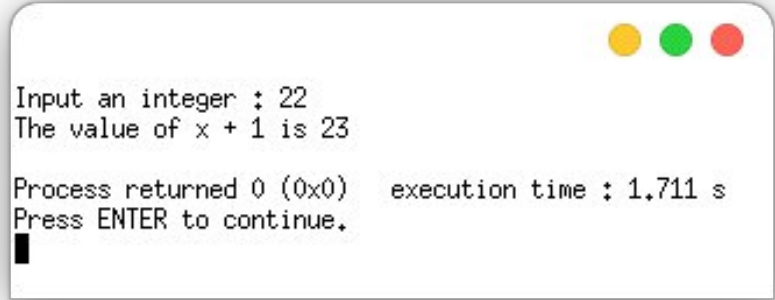


Code :

```
/* This program is written by Vishal Das  
(2K21/A8/24) on May 05, 2022 02:15 PM  
for Lab File of course C0102  
*/  
  
#include <stdio.h>  
  
int main(void){  
    int x;  
    printf("Input an integer : ");  
    scanf("%d", &x);  
    x = x + 1;  
    printf("The value of x + 1 is %d\n", x);  
    return 0;  
}
```

VISHAL DAS
2K21/A8/24

Output :



```
Input an integer : 22
The value of x + 1 is 23

Process returned 0 (0x0)   execution time : 1.711 s
Press ENTER to continue.
█
```

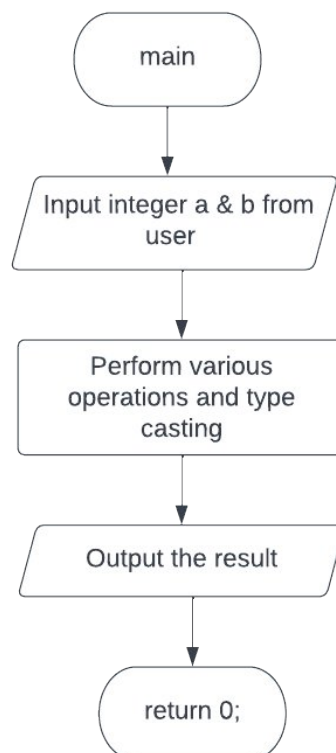
Program 03 : Write a C program to add, subtract, multiplication and division on two integers and print the result and use explicit type casting for division operation.

Theory :

Operators are used to perform various operations of addition, subtraction, multiplication and division which correspond literally to their respective mathematical operators.

Type Casting is a process in which we change a variable belonging to one data type to another one. The Implicit type casting occurs automatically while in Explicit type casting, the programmer needs to force the conversion.

Algorithm :



Code :

```
/* This program is written by Vishal Das  
(2K21/A8/24) on May 05, 2022 02:30 PM  
for Lab File of course C0102  
*/
```

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

VISHAL DAS
2K21/A8/24


```

int main(void)
{
    int a, b, c;
    float d;

    printf("Enter two integers : ");
    scanf("%d %d", &a, &b);

    c = a + b; // Addition Operation
    printf("Addition \t(%d + %d)\t= %d\n", a, b, c);

    c = a - b; // Subtraction Operation
    printf("Subtraction \t(%d - %d)\t= %d\n", a, b, c);

    c = a * b; // Multiplication Operation
    printf("Multiplication \t(%d * %d)\t= %d\n", a, b, c);

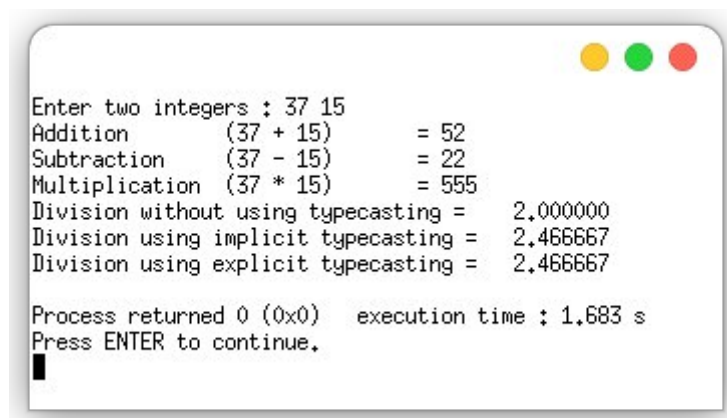
    d = a / b; // Division without using typecasting
    printf("Division without using typecasting = \t%f\n", d);

    d = 1.0 * a / b; // Division using implicit typecasting
    printf("Division using implicit typecasting = \t%f\n", d);

    d = (float)(a) / b; // Division using explicit typecasting
    printf("Division using explicit typecasting = \t%f\n", d);
    return 0;
}

```

Output :



```

Enter two integers : 37 15
Addition      (37 + 15)      = 52
Subtraction   (37 - 15)      = 22
Multiplication (37 * 15)      = 555
Division without using typecasting = 2.000000
Division using implicit typecasting = 2.466667
Division using explicit typecasting = 2.466667

Process returned 0 (0x0)   execution time : 1.683 s
Press ENTER to continue.
█

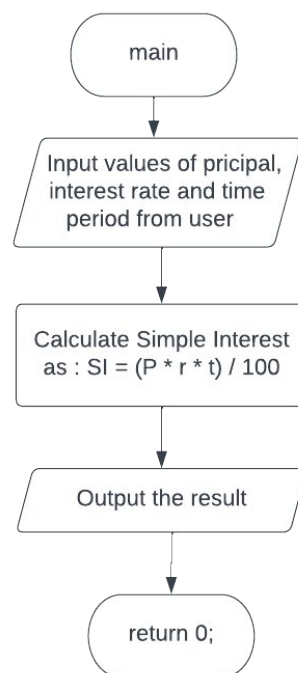
```

Program 04 : Write a C program to calculate simple interest and print on the screen.

Theory :

Simple interest is the interest amount for a particular principal amount of money at some rate of interest calculated by multiplying the annual interest rate by the principal, by the time period in years. Hence, we only need the basic operators of multiplication and division to be used to write a program to calculate SI.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 05, 2022 02:55 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>

```

```

int main(void){
    float principal, rate, time_period, simple_interest;

    printf("Enter the principal amount: ");
    scanf("%f", &principal);

    printf("Enter the rate of interest: ");

```

VISHAL DAS
2K21/A8/24

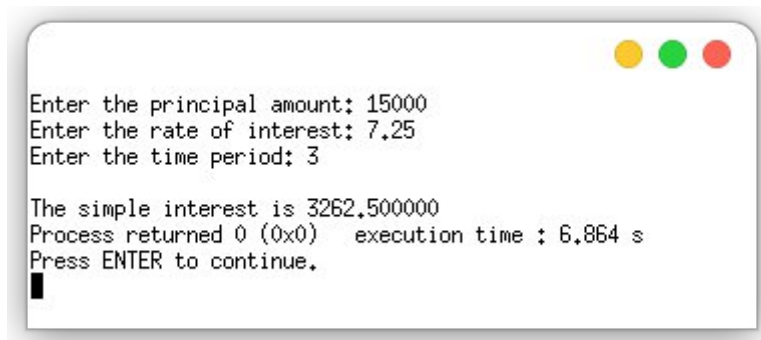
```
scanf("%f", &rate);

printf("Enter the time period: ");
scanf("%f", &time_period);

simple_interest = (principal * rate * time_period) / 100;

printf("\nThe simple interest is %f", simple_interest);
return 0;
}
```

Output :



```
Enter the principal amount: 15000
Enter the rate of interest: 7.25
Enter the time period: 3

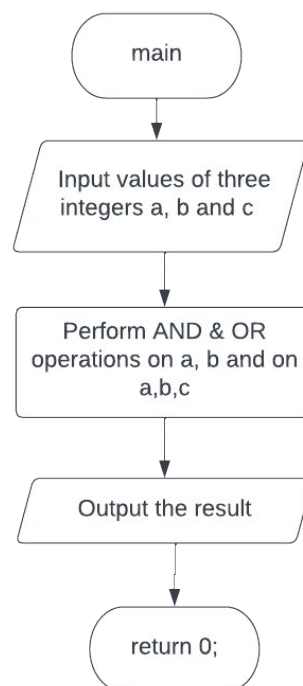
The simple interest is 3262.500000
Process returned 0 (0x0)   execution time : 6.864 s
Press ENTER to continue.
█
```

Program 05 : Write a C program to perform Logical AND and OR operations on two and three variables.

Theory :

The logical operators **&&** and **||** are used when evaluating two expressions to obtain a single relational result. The operator **&&** corresponds to the Boolean logical operation AND, which yields true if both its operands are true, and false otherwise. The operator **||** corresponds to the Boolean logical operation OR, which yields true if either of its operands is true, thus being false only when both operands are false.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 12, 2022 02:10 PM
for Lab File of course C0102
*/

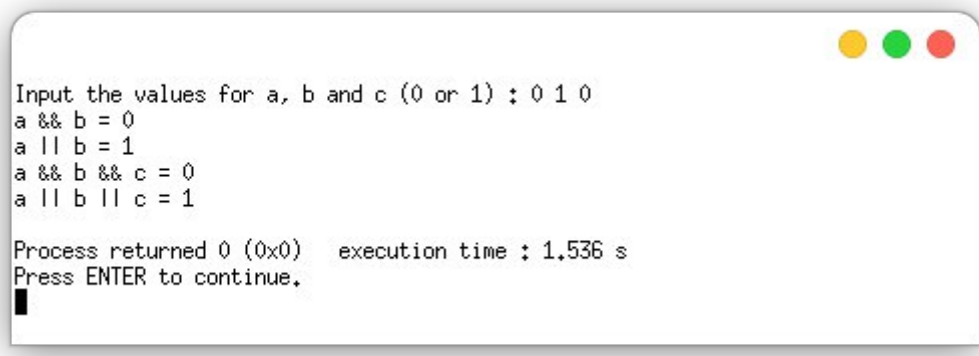
#include <stdio.h>

int main(void){
    int a = 1, b = 0, c = 0;

    printf("Input the values for a, b and c (0 or 1) : ");
    scanf("%d %d %d", &a, &b, &c);
    printf("a && b = %d\n", a && b);
  
```

VISHAL DAS
2K21/A8/24

```
printf("a || b = %d\n", a || b);  
printf("a && b && c = %d\n", a && b && c);  
printf("a || b || c = %d\n", a || b || c);  
  
return 0;  
}
```

Output :

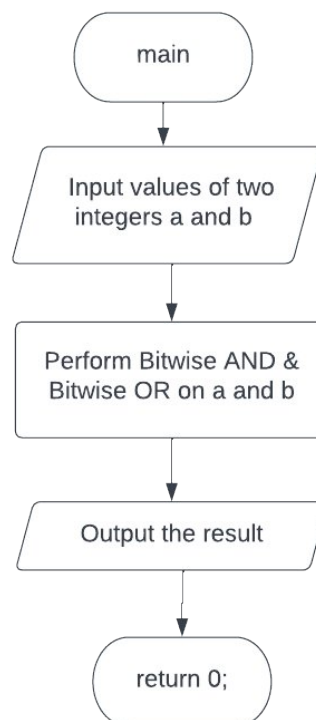
```
Input the values for a, b and c (0 or 1) : 0 1 0  
a && b = 0  
a || b = 1  
a && b && c = 0  
a || b || c = 1  
  
Process returned 0 (0x0)   execution time : 1.536 s  
Press ENTER to continue.  
█
```

Program 06 : Write a C program to perform Bitwise AND and OR operations on two variables.

Theory :

Bitwise operators modify variables considering the bit patterns that represent the values they store. Bitwise AND is performed using operator "&" while Bitwise OR is performed using operator "|".

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 12, 2022 02:45 PM
for Lab File of course C0102
*/

#include <stdio.h>

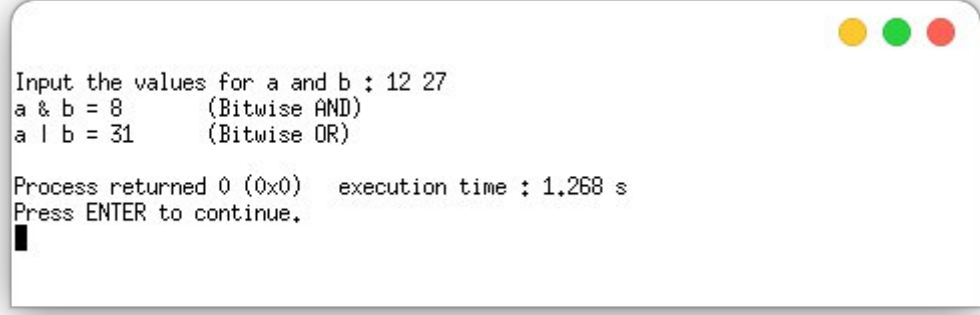
int main(void)
{
    int a, b;

    printf("Input the values for a and b : ");
    scanf("%d %d", &a, &b);
    printf("a & b = %d\t(Bitwise AND)\n", a & b);
  
```

VISHAL DAS
2K21/A8/24

```
printf("a | b = %d\t(Bitwise OR)\n", a | b);  
  
return 0;  
}
```

Output :



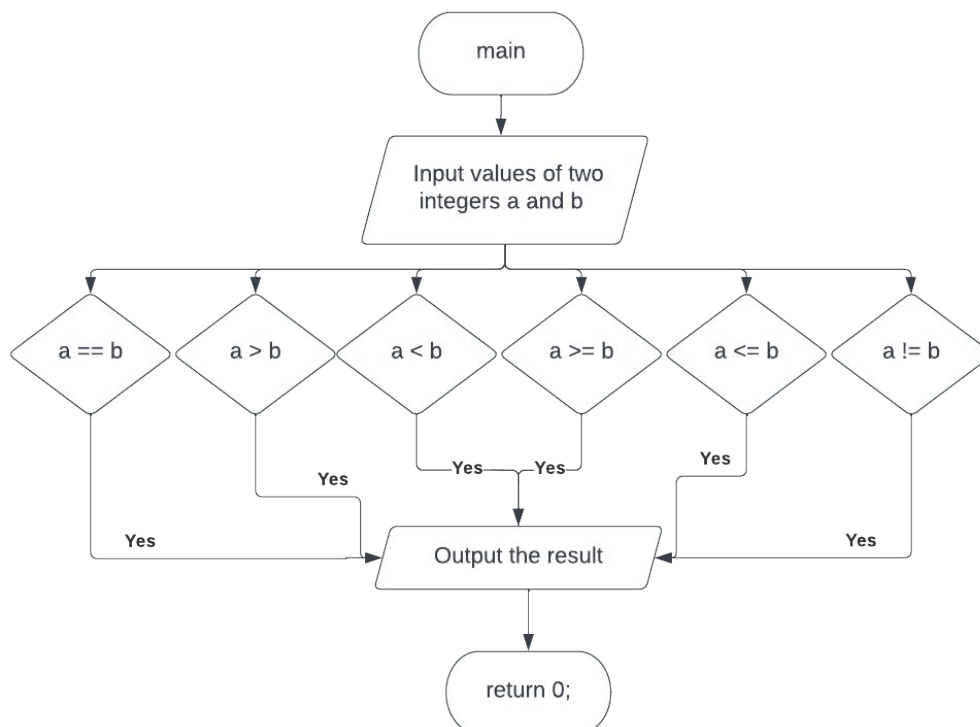
```
Input the values for a and b : 12 27  
a & b = 8      (Bitwise AND)  
a | b = 31     (Bitwise OR)  
  
Process returned 0 (0x0)   execution time : 1.268 s  
Press ENTER to continue.  
█
```

Program 07 : Write a C program to perform relational operations on two variables.

Theory :

Two expressions can be compared using relational and equality operators. The result of such an operation is either true or false (Boolean value). The relational operators available are == (equal to), != (not equal to), > (greater than), < (smaller than), >= (greater than or equal to) and <= (smaller than or equal to)

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 12, 2022 03:15 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>

```

```

int main(void){
    int a=5, b=6;

    printf("Input values of a and b to compare : ");
    scanf("%d %d", &a, &b);

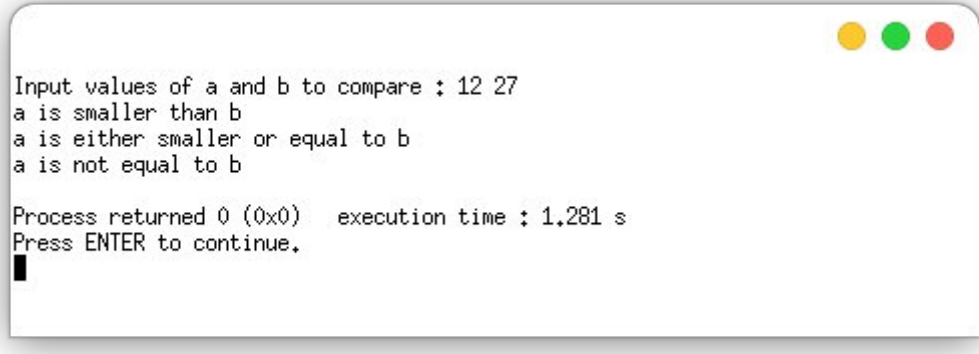
```



```
if(a==b)
    printf("a is equal to b\n");
if(a>b)
    printf("a is greater than b\n");
if(a<b)
    printf("a is smaller than b\n");
if(a<=b)
    printf("a is either smaller or equal to b\n");
if(a>=b)
    printf("a is either greater or equal to b\n");
if(a!=b)
    printf("a is not equal to b\n");

return 0;
}
```

Output :

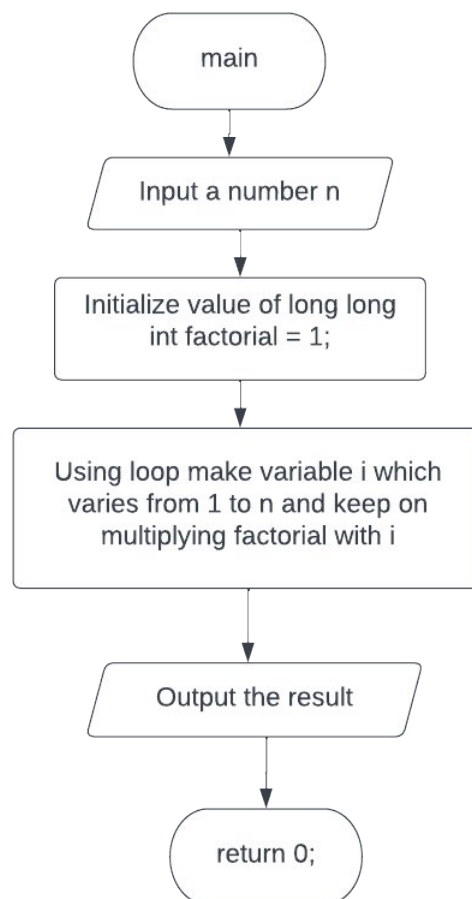


```
Input values of a and b to compare : 12 27
a is smaller than b
a is either smaller or equal to b
a is not equal to b

Process returned 0 (0x0)   execution time : 1.281 s
Press ENTER to continue.
█
```

Program 08 : Write a C program to compute factorial of a number.**Theory :**

Factorial is the product of all positive integers less than or equal to a given positive integer and denoted by that integer and an exclamation point.

Algorithm :**Code :**

```
/* This program is written by Vishal Das  
(2K21/A8/24) on May 19, 2022 02:10 PM  
for Lab File of course C0102  
*/
```

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

```
int main(void){  
    long long int factorial=1;  
    int n;  
  
    printf("Input an integer : ");
```

VISHAL DAS
2K21/A8/24

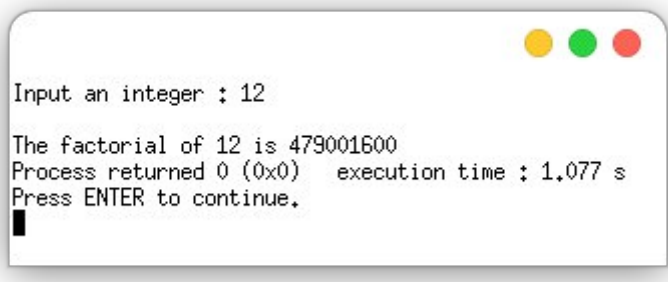
```
scanf("%d", &n);

for(int i=1;i<=n;i++)
    factorial *= i;

printf("\nThe factorial of %d is %d", n, factorial);

return 0;
}
```

Output :



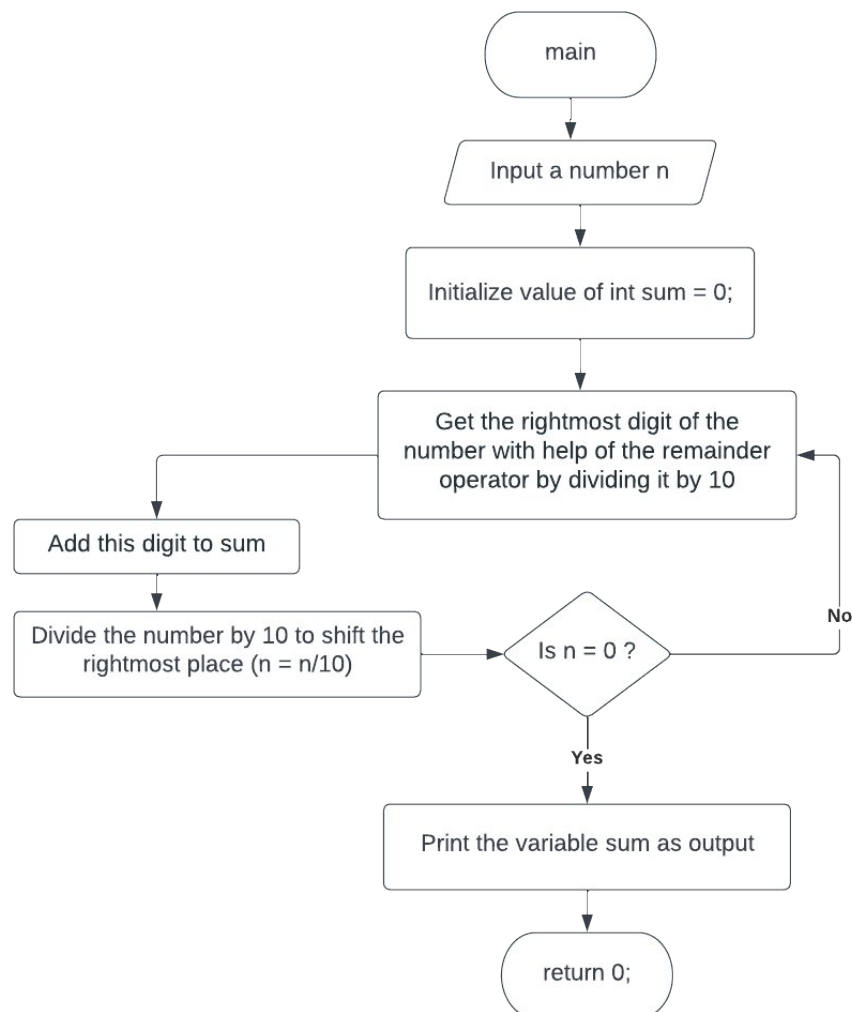
```
Input an integer : 12
The factorial of 12 is 479001600
Process returned 0 (0x0) execution time : 1.077 s
Press ENTER to continue.
█
```

Program 09 : Write a C program to compute sum of digits of a 5-digit number.

Theory :

Sum of digits of a number can be calculated by calculating the sum of rightmost digit of number each time shifting it one place to the right till the number exhausts or becomes zero.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 19, 2022 02:40 PM
for Lab File of course C0102
*/

```

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

```
int main(void){
```

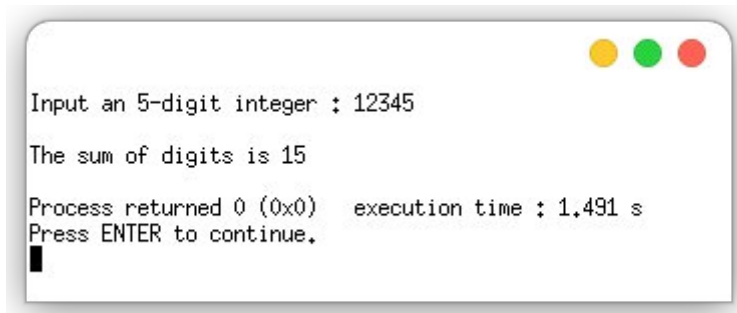
```

VISHAL DAS
2K21/A8/24

```

```
int n;  
int sum = 0;  
  
printf("Input an 5-digit integer : ");  
scanf("%d", &n);  
  
while (n)  
{  
    sum += n % 10;  
    n = n / 10;  
}  
  
printf("\nThe sum of digits is %d\n", sum);  
  
return 0;  
}
```

Output :

A screenshot of a terminal window with a white background and a grey border. In the top right corner, there are three colored window control buttons: yellow, green, and red. The terminal displays the following text: "Input an 5-digit integer : 12345", "The sum of digits is 15", "Process returned 0 (0x0) execution time : 1.491 s", and "Press ENTER to continue." followed by a black cursor line.

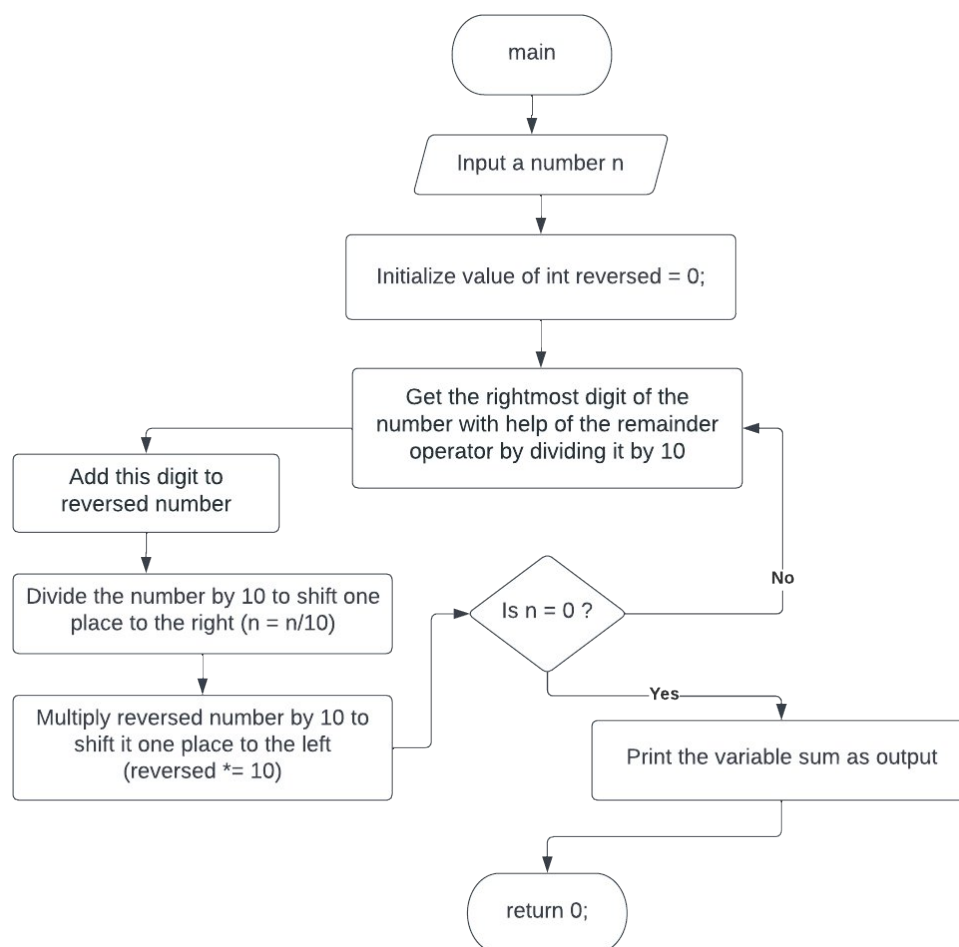
```
Input an 5-digit integer : 12345  
The sum of digits is 15  
Process returned 0 (0x0)   execution time : 1.491 s  
Press ENTER to continue.  
█
```

Program 10 : Write a C program to reverse a 5-digit number.

Theory :

To reverse a number we can add rightmost digit of the original number each time shifting reversed number one place to left and original number one place to the right till the number exhausts or becomes zero.

Algorithm :



Code:

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 19, 2022 03:10 PM
for Lab File of course C0102
*/

```

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

```
int main(void){
    int n;
```

VISHAL DAS
2K21/A8/24

```
int reversed = 0;

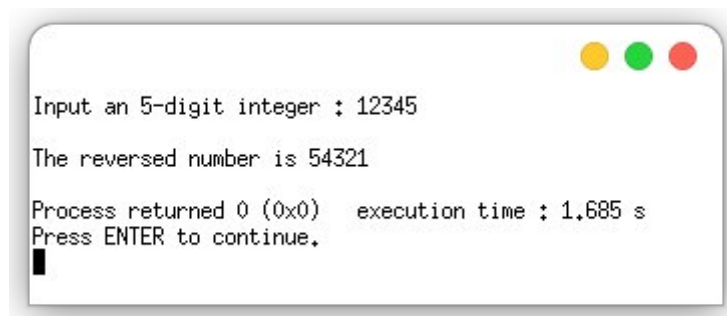
printf("Input an 5-digit integer : ");
scanf("%d", &n);

while (n)
{
    reversed *= 10;
    reversed += n % 10;
    n = n / 10;
}

printf("\nThe reversed number is %d\n", reversed);

return 0;
}
```

Output :

A screenshot of a terminal window with a white background and a grey border. In the top right corner, there are three colored window control buttons: yellow, green, and red. The terminal displays the following text: "Input an 5-digit integer : 12345", "The reversed number is 54321", "Process returned 0 (0x0) execution time : 1.685 s", and "Press ENTER to continue." followed by a black cursor line.

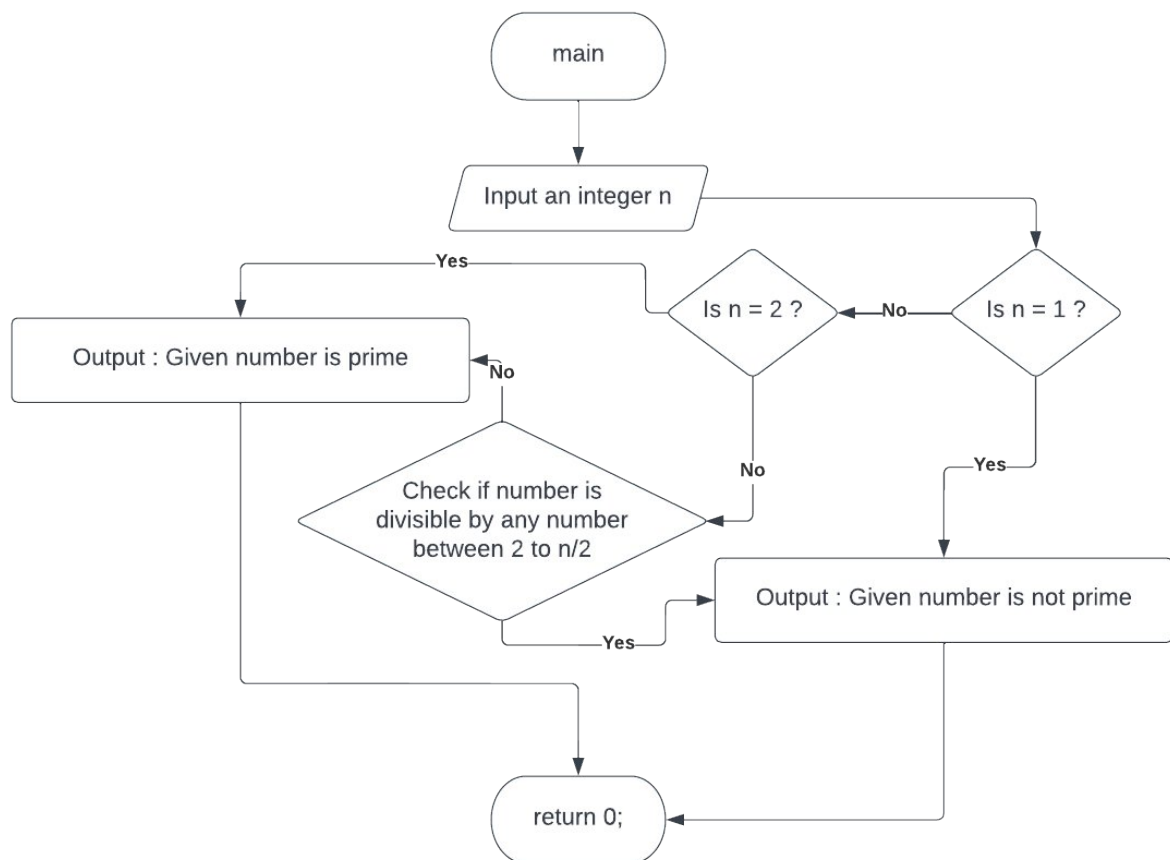
```
Input an 5-digit integer : 12345
The reversed number is 54321
Process returned 0 (0x0)   execution time : 1.685 s
Press ENTER to continue.
█
```

Program 11 : Write a C program to find whether the input number is prime or not.

Theory :

A number is said to be prime if it only have exactly two factors i.e., 1 and the number itself. 1 is not prime because it have only one factor. Other numbers can be checked if prime or not by dividing them with numbers in between 2 and the half of the number because factors repeat after the middle point.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on May 19, 2022 03:30 PM
for Lab File of course C0102
*/

```

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

```

int main(void)
{
    int n;

```

VISHAL DAS
2K21/A8/24

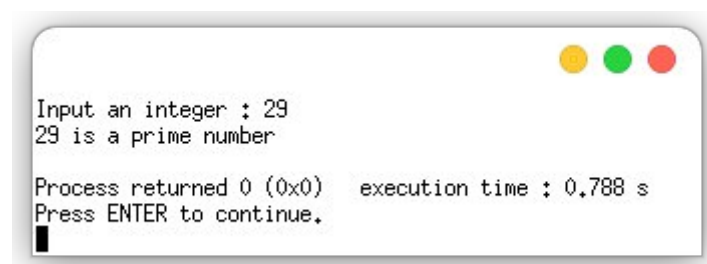

```
printf("Input an integer : ");
scanf("%d", &n);

if (n == 1){
    printf("1 is not a prime number\n", n);
    return 0;
}else if (n == 2){
    printf("2 is a prime number\n", n);
    return 0;
}

for (int factor = 2; factor <= n / 2; factor++){
    if (n % factor == 0){
        printf("%d is not a prime number\n", n);
        return 0;
    }

printf("%d is a prime number\n", n);
return 0;
}
```

Output :



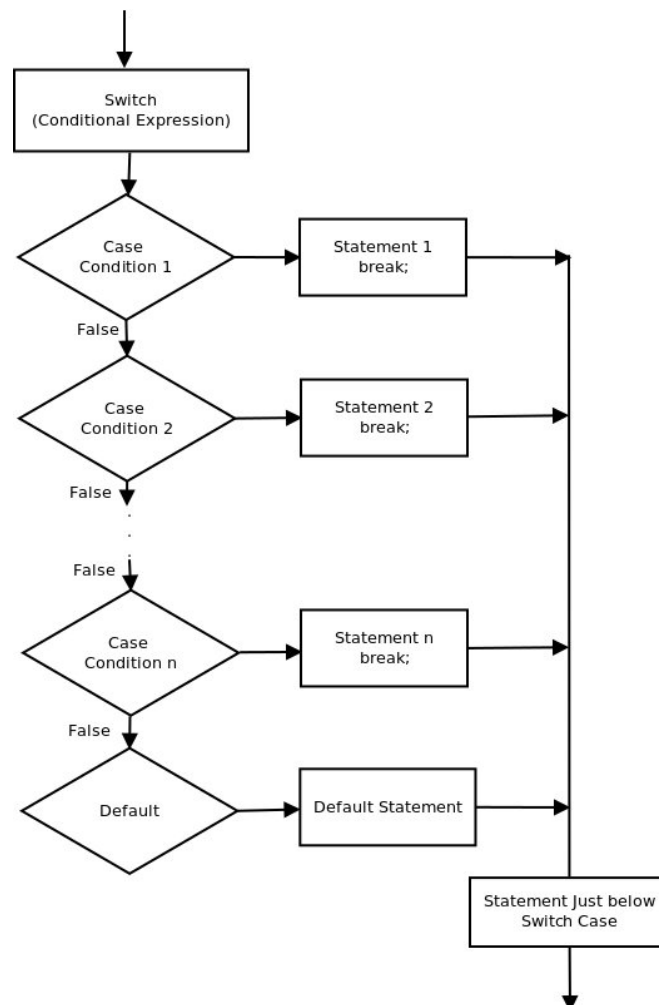
```
Input an integer : 29
29 is a prime number

Process returned 0 (0x0)   execution time : 0.788 s
Press ENTER to continue.
█
```

Program 12 : Write a program to find area of different shapes using switch case.

Theory : Switch-Case purpose is to check for a value among a number of possible constant expressions. It is something similar to concatenating if-else statements, but limited to constant expressions. If the value of expression did not match any of the previously specified constants (there may be any number of these), the program executes the statements included after the default: label, if it exists.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on June 02, 2022 02:00 PM
for Lab File of course C0102
*/
#include <stdio.h>

```

```

int main(void){
    char shape;
    float area;

```

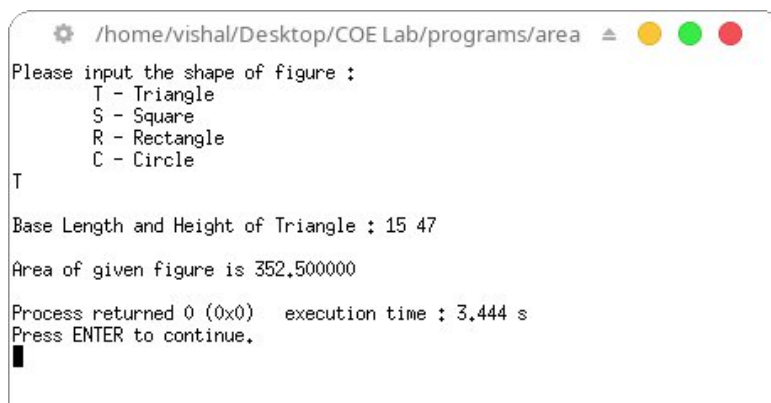
VISHAL DAS
2K21/A8/24

```

    printf("Please input the shape of figure :\n\tT - Triangle\n\tS - Square\n\tR - Rectangle\n\tC - Circle\n");
    scanf("%c", &shape);
    printf("\n");
    switch(shape){
    case 'T':
        float b, h;
        printf("Base Length and Height of Triangle : ");
        scanf("%f %f", &b, &h);
        area = 0.5 * b * h;
        break;
    case 'S':
        float side;
        printf("Side length of Square : ");
        scanf("%f", &side);
        area = side * side;
        break;
    case 'R':
        float l, w;
        printf("Base Length and Width of Rectangle : ");
        scanf("%f %f", &l, &w);
        area = l * w;
        break;
    case 'C':
        float r;
        printf("Radius of Circle : ");
        scanf("%f", &r);
        area = 3.1415 * r * r;
        break;
    default:
        printf("Invalid shape"); return 0;
    }
    printf("\nArea of given figure is %f\n", area);
    return 0;
}

```

Output :



```

/home/vishal/Desktop/COE Lab/programs/area
Please input the shape of figure :
    T - Triangle
    S - Square
    R - Rectangle
    C - Circle
T
Base Length and Height of Triangle : 15 47
Area of given figure is 352.500000
Process returned 0 (0x0)   execution time : 3.444 s
Press ENTER to continue.

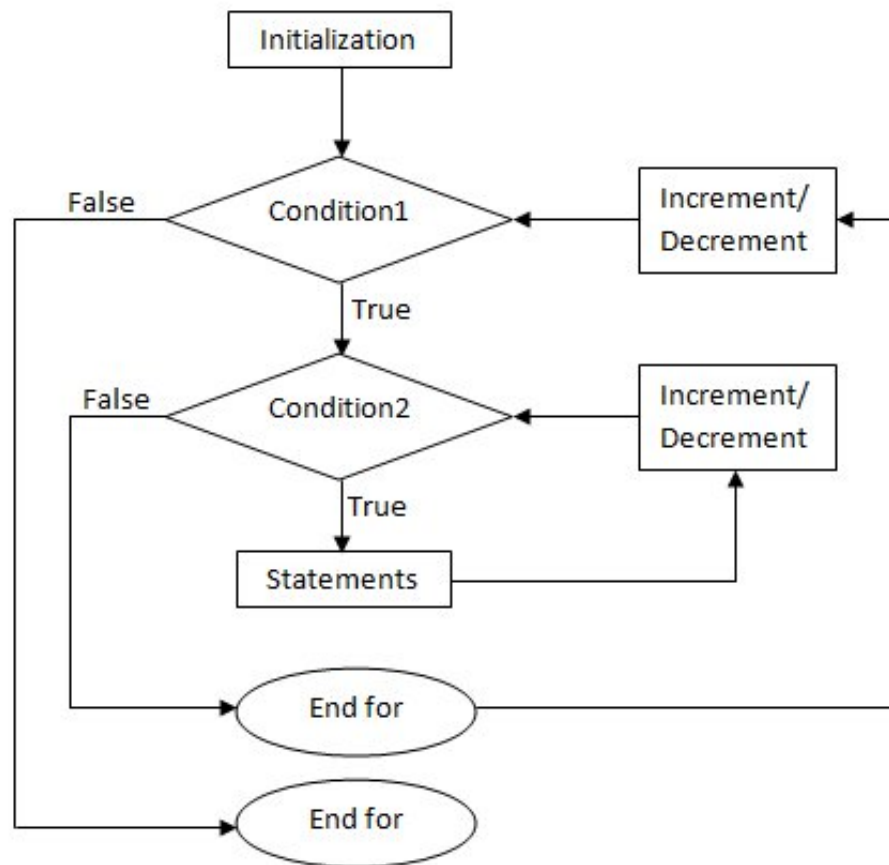
```

Program 13 : Write a program to implement following pattern

1. Half Pyramid of *
2. Half Pyramid of Numbers
3. Inverted Half Pyramid of *
4. Inverted Half Pyramid of Numbers
5. Full Pyramid of *

Theory : Nesting of loops is the feature in C that allows the looping of statements inside another loop. If a loop exists inside the body of another loop, it's called a nested loop. We can use the nested loop to create patterns like full pyramid, half pyramid, inverted pyramid, and so on.

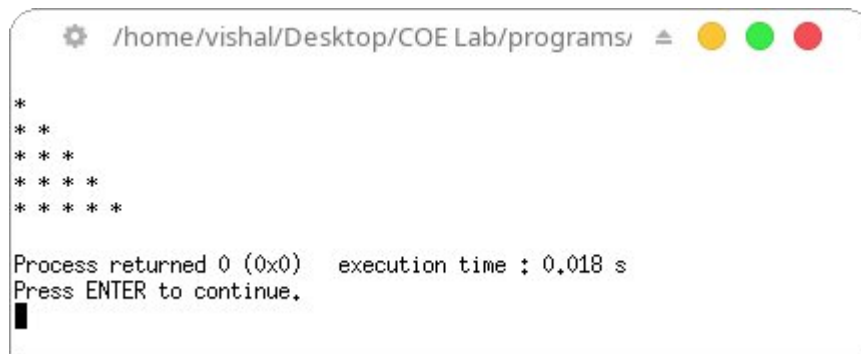
Algorithm :



Code and Output :

1. Half pyramid of *

```
#include <stdio.h>
int main(void){
    int n = 5;
    for(int i=0;i<=n;i++){
        for(int j=1;j<=i;j++){
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

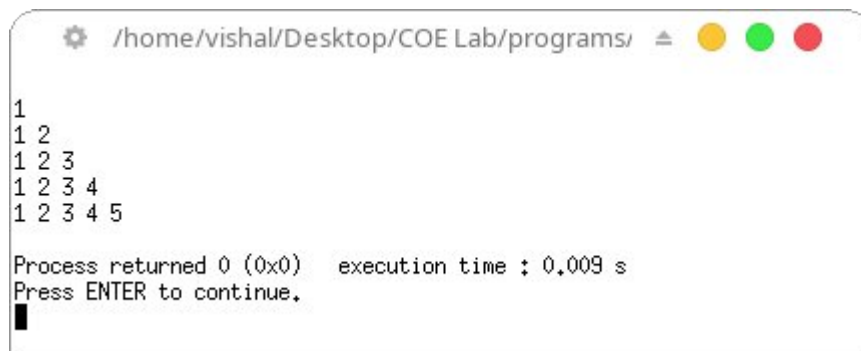


```
/home/vishal/Desktop/COE Lab/programs/
*
* *
* * *
* * * *
* * * * *

Process returned 0 (0x0)   execution time : 0.018 s
Press ENTER to continue.
```

2. Half Pyramid of Numbers

```
#include <stdio.h>
int main(void){
    int n = 5;
    for(int i=0;i<=n;i++){
        for(int j=1;j<=i;j++){
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

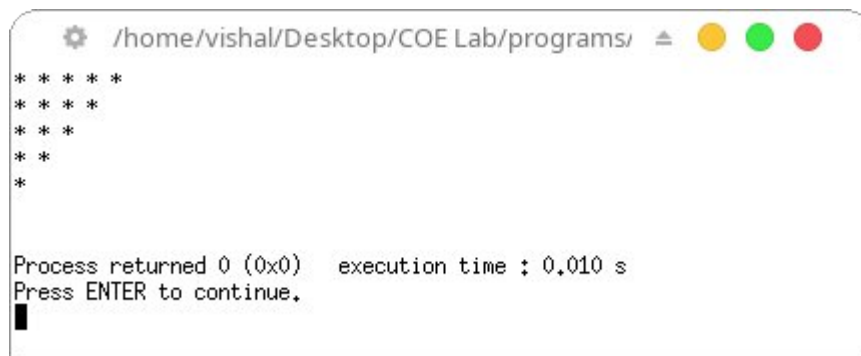


```
/home/vishal/Desktop/COE Lab/programs/
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Process returned 0 (0x0)   execution time : 0.009 s
Press ENTER to continue.
```

3. Inverted Half Pyramid of *

```
#include <stdio.h>
int main(void){
    int n = 5;
    for(int i=0;i<=n;i++){
        int k = n-i;
        for(int j=1;j<=k;j++){
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

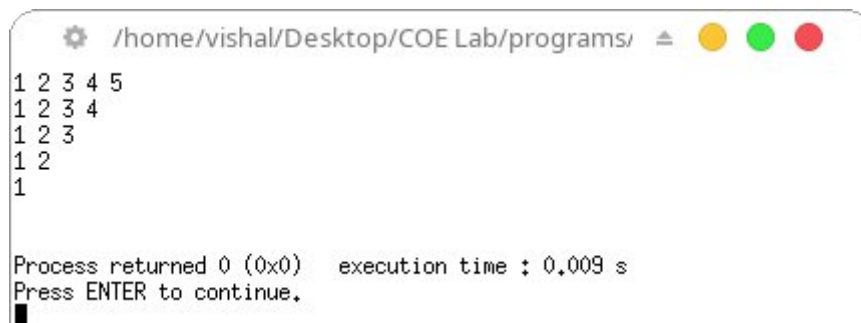


```
/home/vishal/Desktop/COE Lab/programs/
* * * * *
* * * *
* * *
* *
*

Process returned 0 (0x0)   execution time : 0.010 s
Press ENTER to continue.
```

4. Inverted Half Pyramid of Numbers

```
#include <stdio.h>
int main(void){
    int n = 5;
    for(int i=0;i<=n;i++){
        int k = n-i;
        for(int j=1;j<=k;j++){
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```



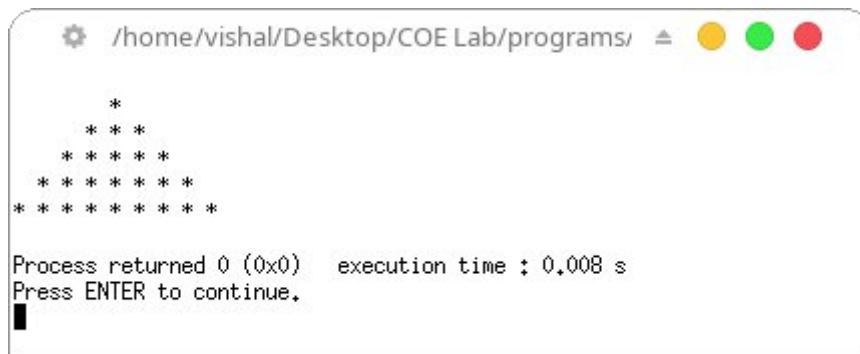
```
/home/vishal/Desktop/COE Lab/programs/
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

Process returned 0 (0x0)   execution time : 0.009 s
Press ENTER to continue.
```

5. Full Pyramid of *

```
#include <stdio.h>

int main(void){
    int n = 5;
    int k;
    for(int i=0;i<=n;i++){
        k = n-i;
        for(int j=1;j<=k;j++){
            printf(" ");
        }
        k = 2*i-1;
        for(int j=1;j<=k;j++){
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```



```
/home/vishal/Desktop/COE Lab/programs/
    *
   ***
  *****
 *****
*****

Process returned 0 (0x0)   execution time : 0.008 s
Press ENTER to continue.
█
```

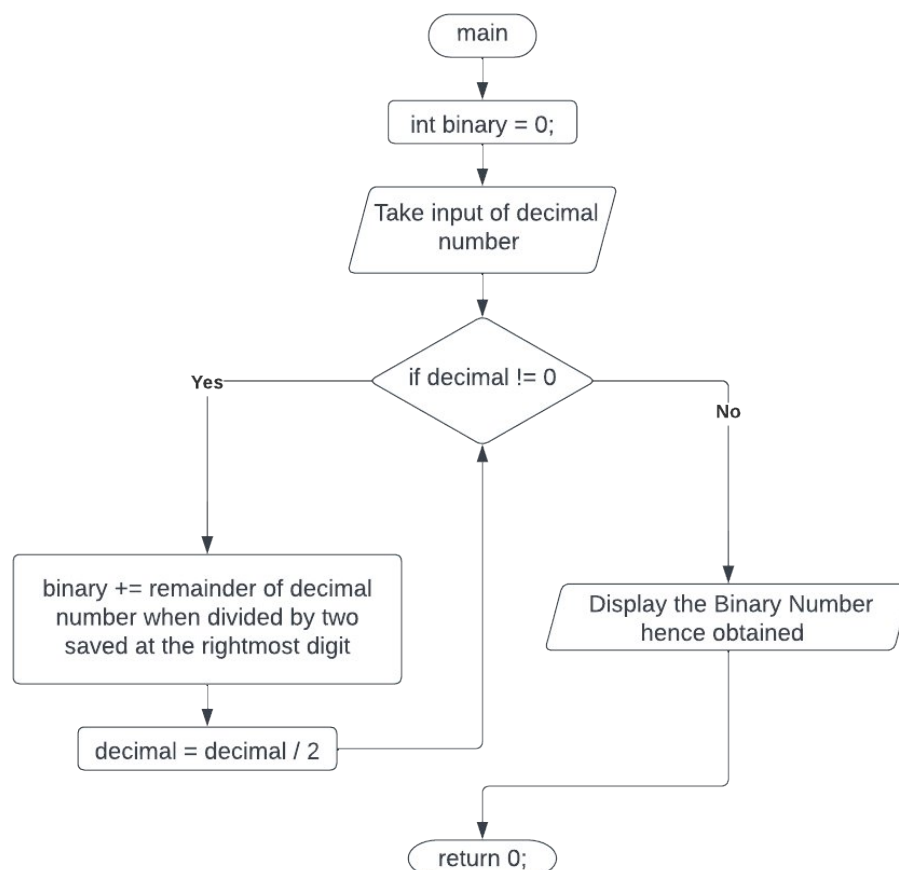
Program 14 : Write a program to convert

1. Decimal to Binary Number
2. Binary to Decimal Number

Theory : Binary numbers are composed of only 0 and 1, whereas, decimal numbers are composed of digits from 0 to 9. The binary number system is also called the base-2 number system and the decimal number system is known as the base-10 number system. A number given in the binary number system can be converted to its equivalent in the decimal number system and vice versa by appropriate algorithm as discussed.

1. Decimal to Binary Number

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on June 02, 2022 03:00 PM
for Lab File of course C0102
*/

```

```

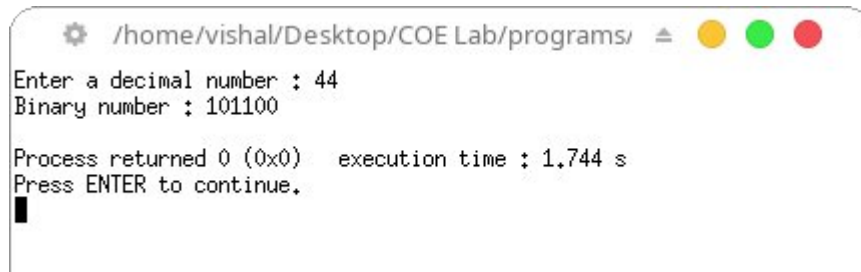
#include <math.h>
#include <stdio.h>

```



```
int main(void)
{
    int decimal, binary = 0, i = 0;
    printf("Enter a decimal number : ");
    scanf("%d", &decimal);
    while (decimal != 0)
    {
        binary += decimal % 2 * pow(10, i++);
        decimal /= 2;
    }
    printf("Binary number : %d\n", binary);
    return 0;
}
```

Output :

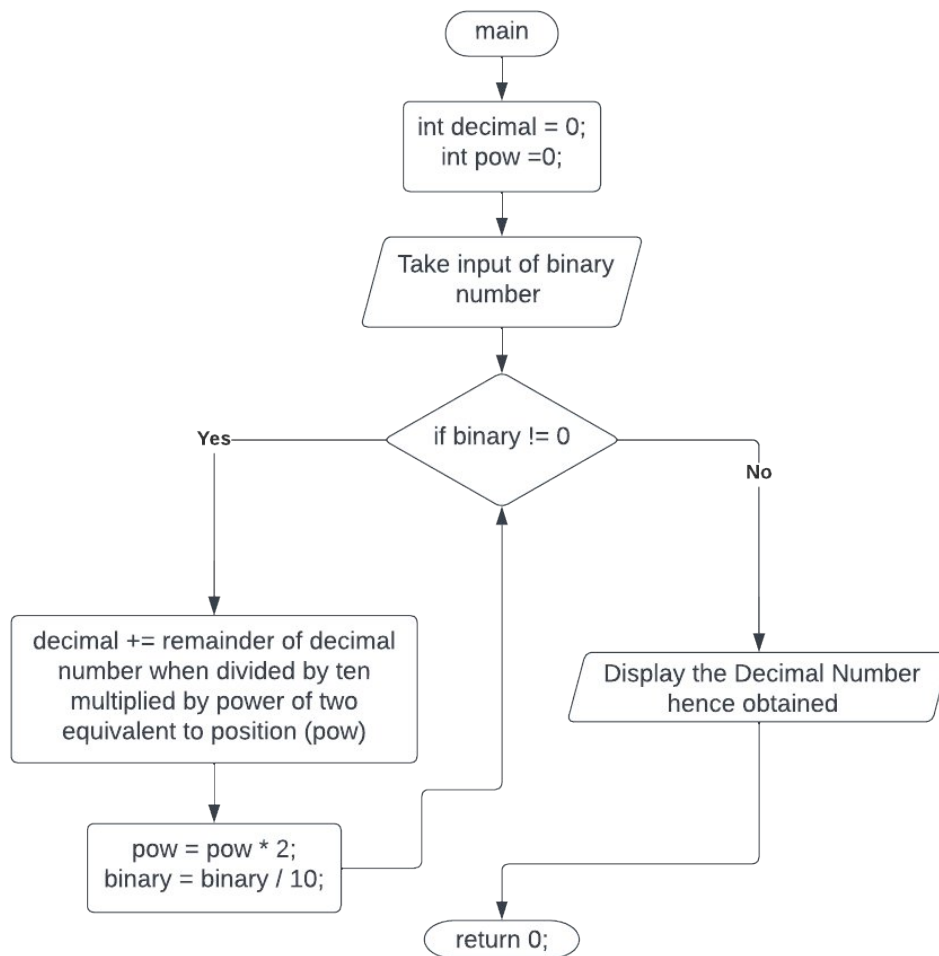


```
/home/vishal/Desktop/COE Lab/programs/
Enter a decimal number : 44
Binary number : 101100

Process returned 0 (0x0)   execution time : 1.744 s
Press ENTER to continue.
█
```

1. Binary to Decimal Number

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on June 02, 2022 03:20 PM
for Lab File of course C0102
*/

```

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

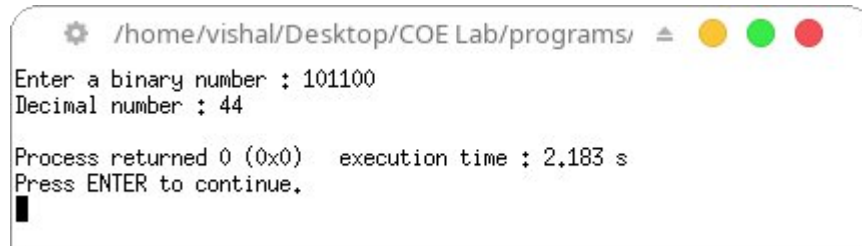
```

int main(void){
    int binary, pow = 1, decimal = 0;
    printf("Enter a binary number : ");
    scanf("%d", &binary);
    while (binary != 0){
        decimal += (binary % 10) * pow;
        pow *= 2;
        binary /= 10;
    }
    printf("Decimal number is: %d", decimal);
    return 0;
}

```

```
}  
printf("Decimal number : %d\n", decimal);  
return 0;  
}
```

Output :

A terminal window with a title bar showing the path /home/vishal/Desktop/COE Lab/programs/. The window contains the following text: "Enter a binary number : 101100", "Decimal number : 44", "Process returned 0 (0x0) execution time : 2.183 s", and "Press ENTER to continue." followed by a cursor.

```
/home/vishal/Desktop/COE Lab/programs/  
Enter a binary number : 101100  
Decimal number : 44  
Process returned 0 (0x0) execution time : 2.183 s  
Press ENTER to continue.  
█
```

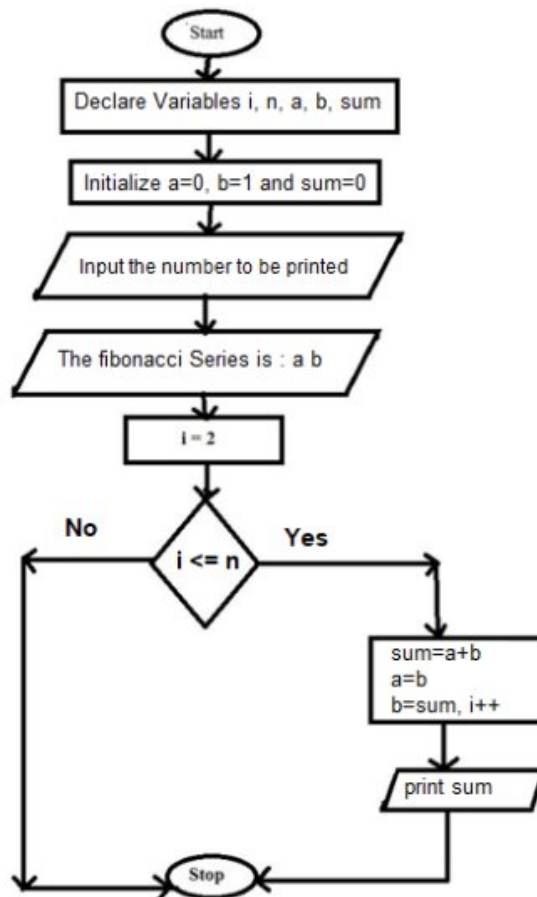
Program 15 : Program to generate the Fibonacci sequence

Theory :

Fibonacci sequence, in which each number is the sum of the two preceding ones. The sequence commonly starts from 0 and 1.

$$F_n = F_{n-1} + F_{n-2}$$

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:00 PM
for Lab File of course C0102
*/

```

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

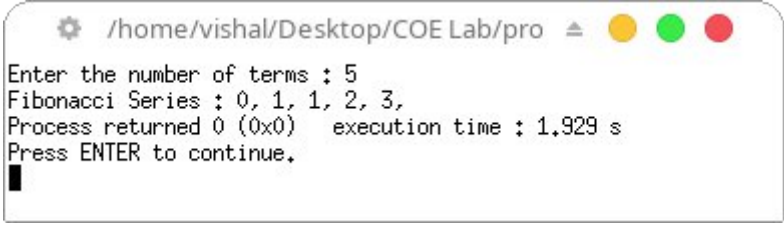
```

int main() {
    int i, n;
    int t1 = 0, t2 = 1;
    int nextTerm = t1 + t2;

```

```
printf("Enter the number of terms : ");
scanf("%d", &n);
printf("Fibonacci Series : %d, %d, ", t1, t2);
for (i = 3; i <= n; ++i) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2;
}
return 0;
}
```

Output :

A terminal window with a title bar showing the path "/home/vishal/Desktop/COE Lab/pro". The window contains the following text: "Enter the number of terms : 5", "Fibonacci Series : 0, 1, 1, 2, 3,", "Process returned 0 (0x0) execution time : 1.929 s", and "Press ENTER to continue." followed by a cursor.

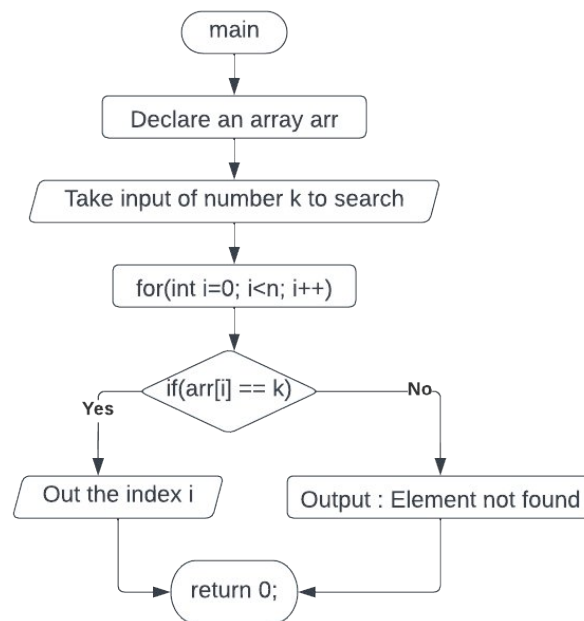
```
/home/vishal/Desktop/COE Lab/pro
Enter the number of terms : 5
Fibonacci Series : 0, 1, 1, 2, 3,
Process returned 0 (0x0) execution time : 1.929 s
Press ENTER to continue.
```

Program 16 : Program to search a number from an array using linear search

Theory :

In a linear search, we start searching an element from the left most element of the array and one by one compare the required element with each element of array

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:10 PM
for Lab File of course C0102
*/

```

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

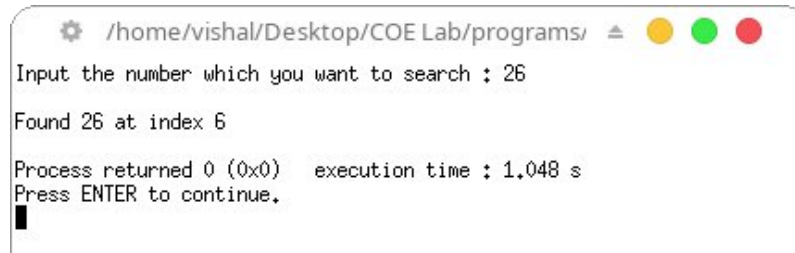
```

int main(void){
    int n;
    int arr[] = {10,6,5,8,9,75,26,64}; // array size = 8
    printf("Input the number which you want to search : ");
    scanf("%d", &n);
    for(int i=0; i<8; i++)
    {
        if(arr[i]==n){
            printf("\nFound %d at index %d\n", arr[i], i);
            return 0;
        }
    }
}

```

```
}  
printf("\nThe number does not exists in the array\n");  
return 0;  
}
```

Output :

A terminal window with a title bar showing the path /home/vishal/Desktop/COE Lab/programs/. The window contains the following text: "Input the number which you want to search : 26", "Found 26 at index 6", "Process returned 0 (0x0) execution time : 1.048 s", and "Press ENTER to continue." followed by a cursor.

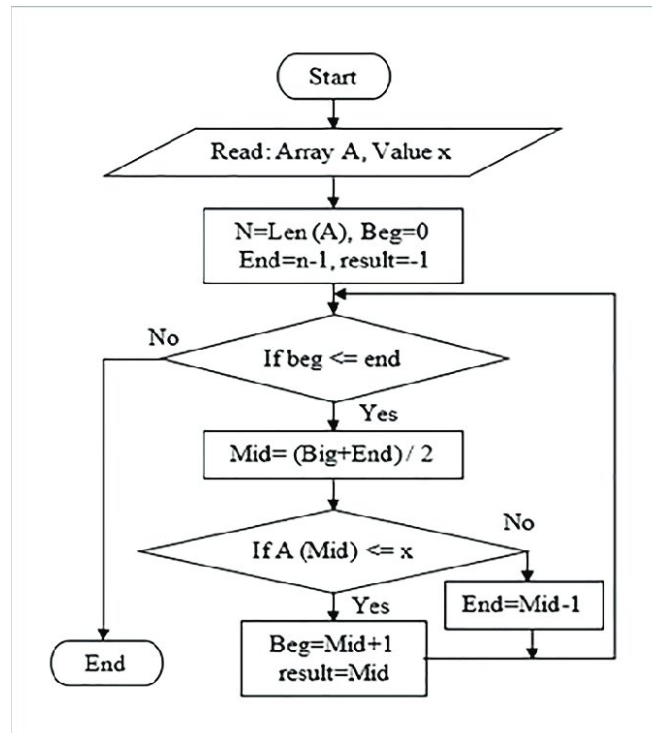
```
/home/vishal/Desktop/COE Lab/programs/  
Input the number which you want to search : 26  
Found 26 at index 6  
Process returned 0 (0x0) execution time : 1.048 s  
Press ENTER to continue.  
█
```

Program 17 : Program to search a number from an array using binary search

Theory :

Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log n)$.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:00 PM
for Lab File of course C0102
*/

```

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

```

int binarySearch(int arr[], int l, int r, int x){
    if (r >= l){
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
}

```



```
        return -1;
    }

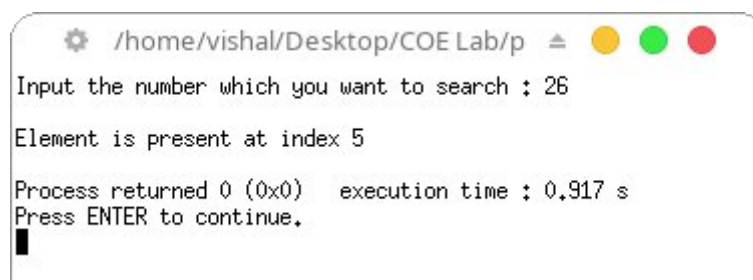
    int main(void){
        int arr[] = {5,6,8,9,10,26,64,75};
        int n = sizeof(arr) / sizeof(arr[0]);
        int x;
        printf("Input the number which you want to search : ");
        scanf("%d", &x);

        int result = binarySearch(arr, 0, n - 1, x);

        if(result == -1){ printf("\nElement is not present in array\n"); }
        else{ printf("\nElement is present at index %d\n", result); }

        return 0;
    }
```

Output :

A terminal window with a title bar showing the path /home/vishal/Desktop/COE Lab/p. The window contains the following text: "Input the number which you want to search : 26", "Element is present at index 5", "Process returned 0 (0x0) execution time : 0.917 s", and "Press ENTER to continue." followed by a cursor.

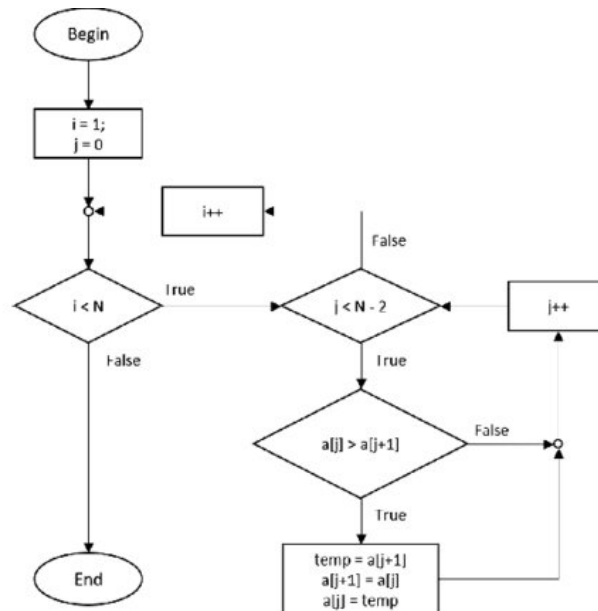
```
/home/vishal/Desktop/COE Lab/p
Input the number which you want to search : 26
Element is present at index 5
Process returned 0 (0x0) execution time : 0.917 s
Press ENTER to continue.
█
```

Program 18 : Program to sort an array using Bubble sort

Theory :

In this sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:30 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>
int main(void){
    int arr[20], num, x, y, temp;
    printf("Please Enter the Number of Elements in the array : ");
    scanf("%d", &num);
    printf("Please Enter the Value of Elements : ");
    for(x = 0; x < num; x++){
        scanf("%d", &arr[x]);
    }
    for(x = 0; x < num - 1; x++){
        for(y = 0; y < num - x - 1; y++){
            if(arr[y] > arr[y + 1]){
                temp = arr[y];
                arr[y] = arr[y + 1];
                arr[y + 1] = temp;
            }
        }
    }
}

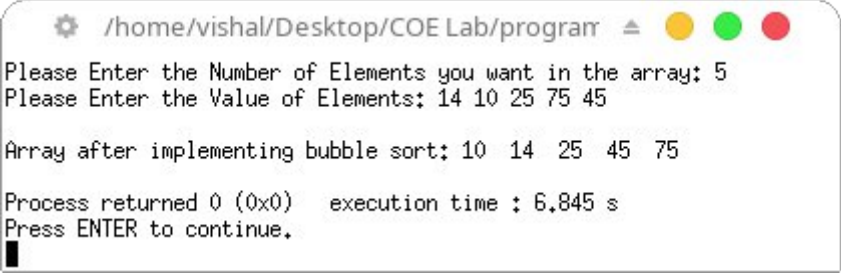
```

```
}

printf("\nArray after implementing bubble sort: ");

for(x = 0; x < num; x++){
    printf("%d  ", arr[x]);
}
printf("\n");
return 0;
}
```

Output :



```
/home/vishal/Desktop/COE Lab/program
Please Enter the Number of Elements you want in the array: 5
Please Enter the Value of Elements: 14 10 25 75 45

Array after implementing bubble sort: 10 14 25 45 75

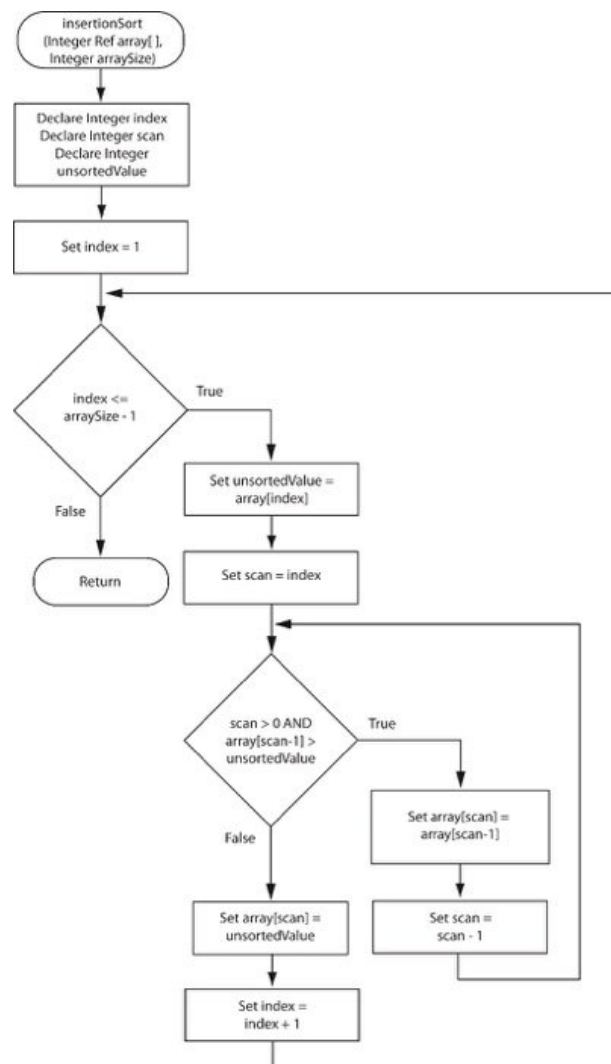
Process returned 0 (0x0)   execution time : 6.845 s
Press ENTER to continue.
```

Program 19 : Program to sort an array using Insertion sort

Theory :

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:10 PM
for Lab File of course C0102
*/

```

```

#include <math.h>
#include <stdio.h>
void insertionSort(int arr[], int n){
    int i, key, j;

```

VISHAL DAS
2K21/A8/24

```

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void printArray(int arr[], int n){
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

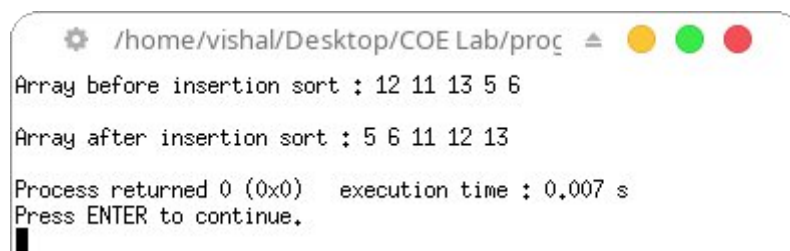
int main(){
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Array before insertion sort : ");
    printArray(arr, n);
    insertionSort(arr, n);
    printf("\nArray after insertion sort : ");
    printArray(arr, n);

    return 0;
}

```

Output :



```

/home/vishal/Desktop/COE Lab/prog
Array before insertion sort : 12 11 13 5 6
Array after insertion sort : 5 6 11 12 13
Process returned 0 (0x0)   execution time : 0.007 s
Press ENTER to continue.

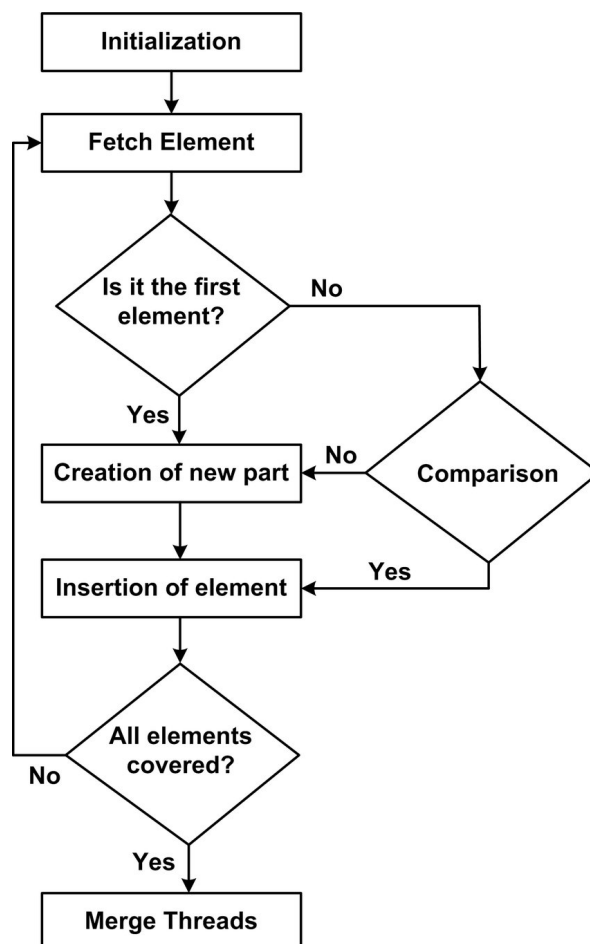
```

Program 20 : Program to sort an array using Selection sort

Theory :

In this sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:10 PM
for Lab File of course C0102
*/

```

```

void swap(int *xp, int *yp){
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

```

```

void selectionSort(int arr[], int n){
    int i, j, min_idx;

```

VISHAL DAS
2K21/A8/24

```

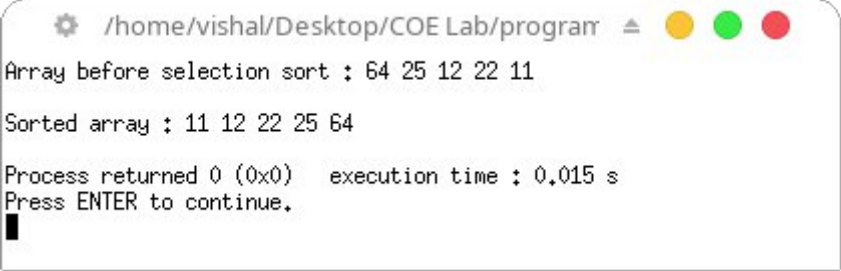
for (i = 0; i < n-1; i++)
{
    min_idx = i;
    for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;
    swap(&arr[min_idx], &arr[i]);
}

void printArray(int arr[], int size){
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main(){
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Array before insertion sort : ");
    printArray(arr, n);
    selectionSort(arr, n);
    printf("\nSorted array : ");
    printArray(arr, n);
    return 0;
}

```

Output :



```

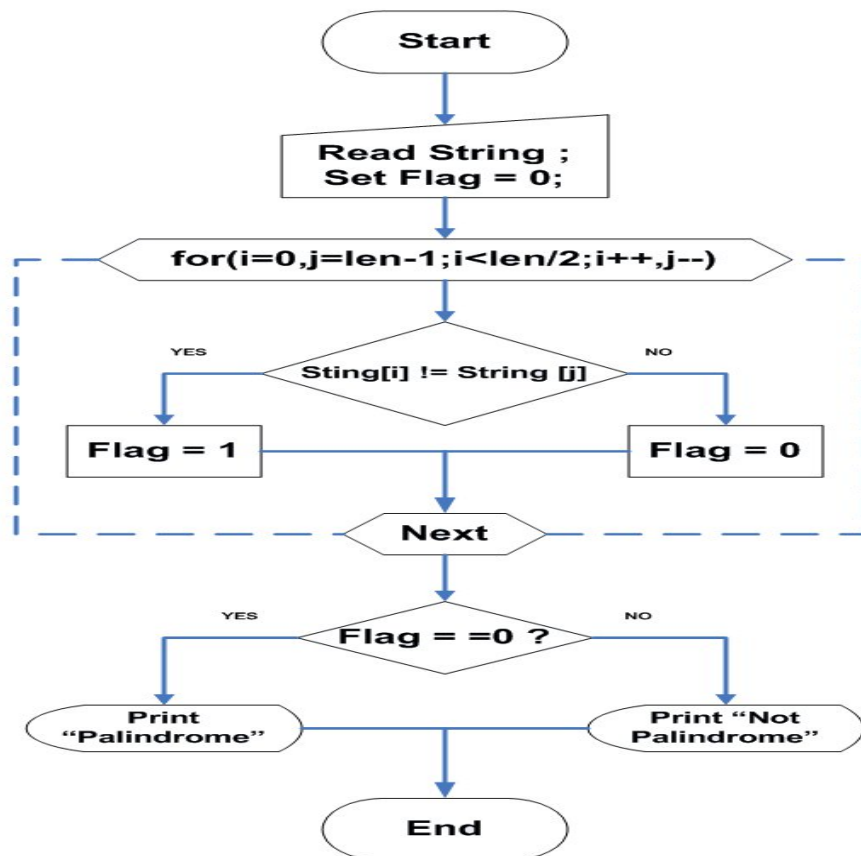
/home/vishal/Desktop/COE Lab/program
Array before selection sort : 64 25 12 22 11
Sorted array : 11 12 22 25 64
Process returned 0 (0x0)   execution time : 0.015 s
Press ENTER to continue.

```

Program 21 : Program to check if a given string is a palindrome or not

Theory : A string is said to be palindrome if it is written the same as original string on reversal.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:15 PM
for Lab File of course C0102
*/
#include <stdio.h>
#include <string.h>

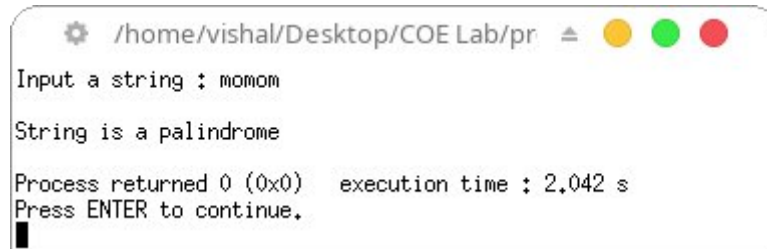
int main(){
    char str[20];
    int len, temp=0;
    printf("Input a string : ");
    scanf("%s", str);
    len = strlen(str);
    for(int i=0; i < len ; i++){
        if(str[i] != str[len-i-1]){
            temp = 1;
        }
    }
}

```



```
        break;
    }
}
if (temp==0){ printf("\nString is a palindrome\n"); }
else{ printf("\nString is not a palindrome\n"); }
return 0;
}
```

Output :

A terminal window with a title bar showing the path /home/vishal/Desktop/COE Lab/pr and standard window control buttons. The terminal displays the following text: 'Input a string : momom', 'String is a palindrome', 'Process returned 0 (0x0) execution time : 2.042 s', and 'Press ENTER to continue.' followed by a cursor.

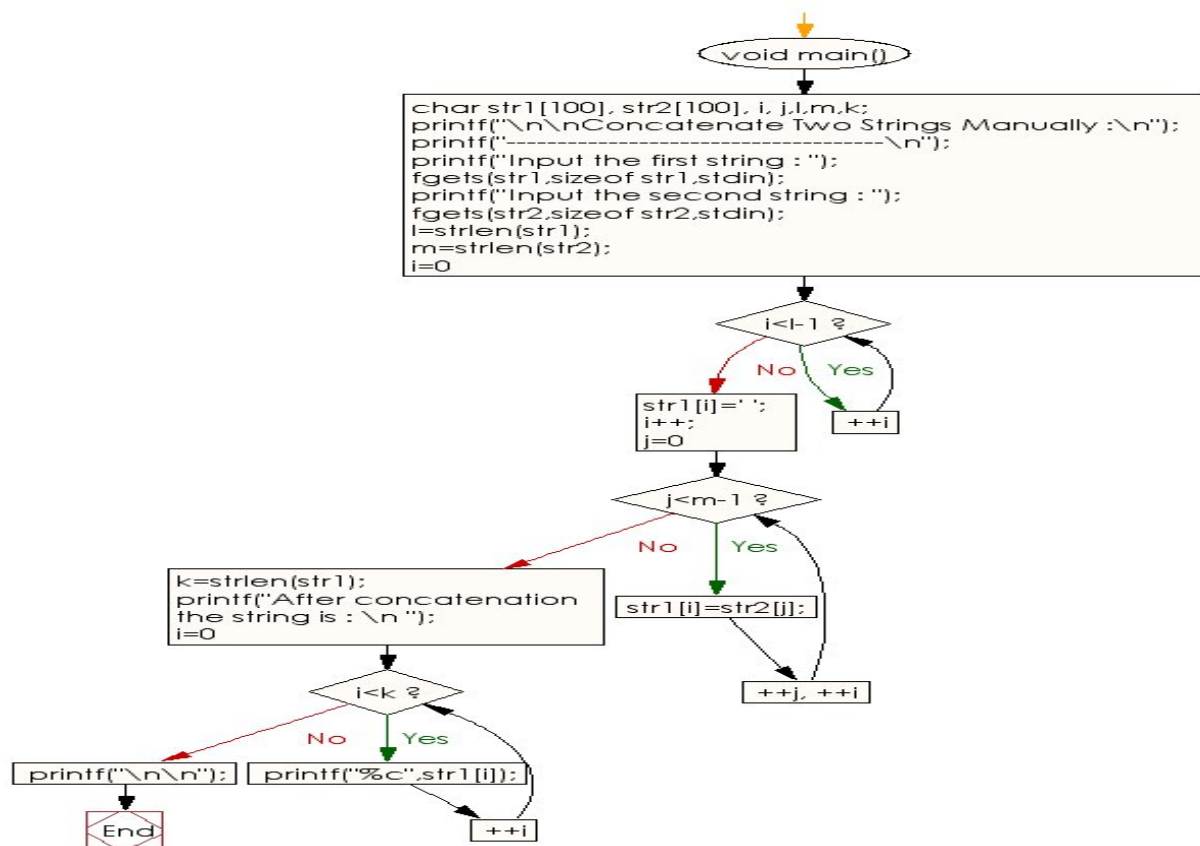
```
/home/vishal/Desktop/COE Lab/pr
Input a string : momom
String is a palindrome
Process returned 0 (0x0) execution time : 2.042 s
Press ENTER to continue.
```

Program 22 : Program to string concatenation

Theory :

In C, string concatenation is the operation of joining character strings end-to- end. It's very easy to use + operator for string concatenation. Strings are immutable, therefore, whenever it is concatenated, it is assigned to a new variable. We can also use % operator for string concatenation. It's useful when we want to concatenate strings and perform simple formatting. Here, the % Operator combine the string that is stored in the string variable. The %s denotes string data type.

Algorithm :



Code :

```

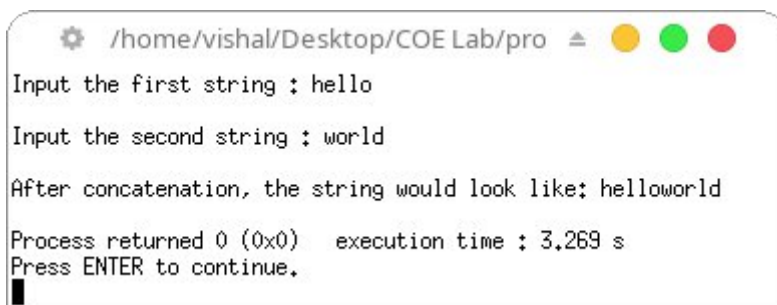
/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:40 PM
for Lab File of course C0102
*/
/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:40 PM
for Lab File of course C0102
*/
  
```

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

VISHAL DAS
2K21/A8/24

```
int main(){
    char first_string[20];
    char second_string[20];
    printf("Input the first string : ");
    scanf("%s",first_string);
    printf("\nInput the second string : ");
    scanf("%s",second_string);
    int i;
    for(i=0; first_string[i]!='\0'; i++);
    for(int j=0; second_string[j]!='\0'; j++){
        first_string[i]=second_string[j];
        i++;
    }
    first_string[i]='\0';
    printf("\nAfter concatenation, the string would look like: %s\n",
first_string);
    return 0;
}
```

Output :



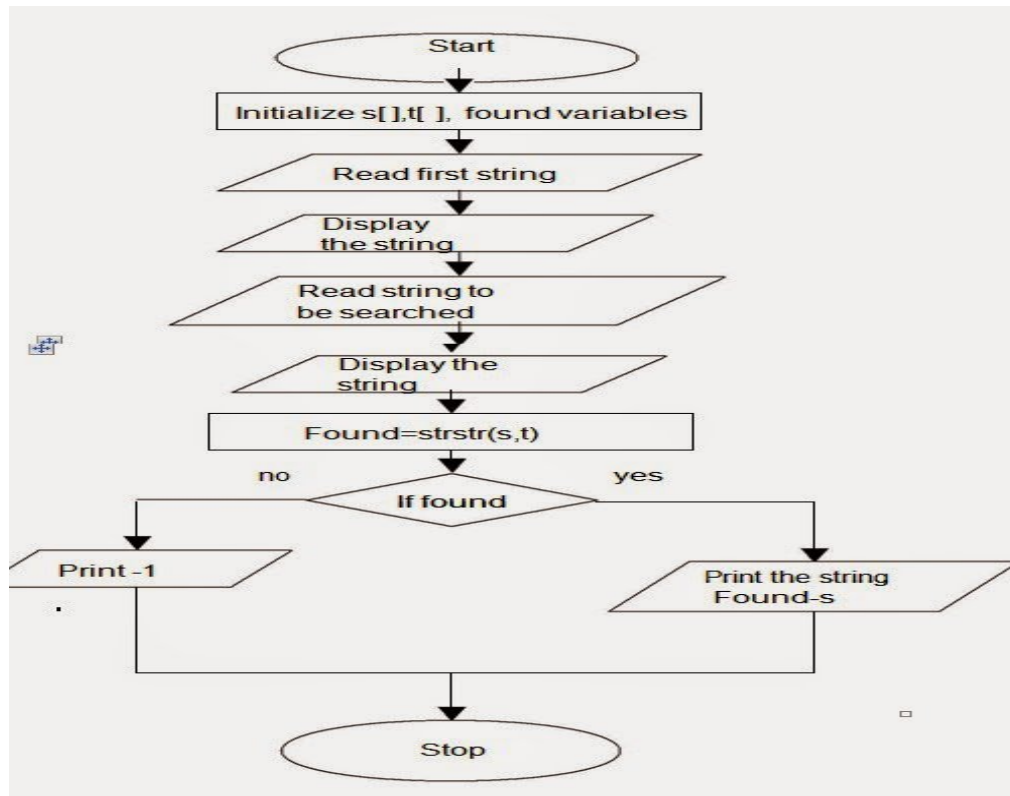
```
/home/vishal/Desktop/COE Lab/pro
Input the first string : hello
Input the second string : world
After concatenation, the string would look like: helloworld
Process returned 0 (0x0)   execution time : 3.269 s
Press ENTER to continue.
```

Program 23 : Program to string comparison

Theory :

Two strings are said to be equal if their string length are equal and ASCII value of each character in both string are equal respectively.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 03:50 PM
for Lab File of course C0102
*/
#include <stdio.h>
#include <string.h>

int str_compare(char* str1, char* str2) {
    int str1_len = strlen(str1);
    int str2_len = strlen(str2);
    if (str1_len != str2_len)
        return -1;
    for (int i = 0; i < str1_len; i++) {
        if (str1[i] != str2[i])
            return -1;
    }
    return 1;
}

```


```

}

int main() {
    char s1[] = "hello world";
    char s2[] = "hello world";
    char s3[] = "hEllo World";
    printf("s1 : %s\ns2 : %s\ns3 : %s", s1, s2, s3);
    printf("\n\ns1 and s2 are ");
    printf((str_compare(s1,s2)==1) ? "equal strings" : "unequal
strings");
    printf("\ns1 and s3 are ");
    printf((str_compare(s2,s3)==1) ? "equal strings" : "unequal
strings");
    printf("\n");
    return 0;
}

```

Output :



```

/home/vishal/Downloads/q23
s1 : hello world
s2 : hello world
s3 : hEllo World

s1 and s2 are equal strings
s1 and s3 are unequal strings

Process returned 0 (0x0)   execution time : 0.005 s
Press ENTER to continue.

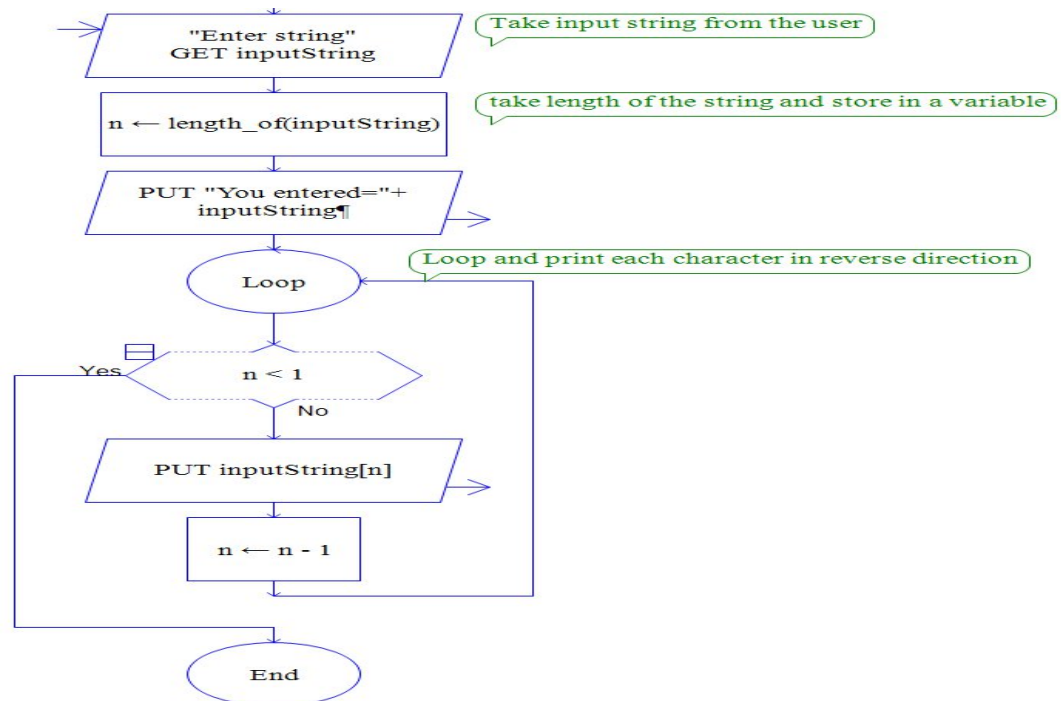
```

Program 24 : Program to reverse string

Theory :

Reversing a string is the technique that reverses or changes the order of a given string so that the last character of the string becomes the first character of the string and so on

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 04:10 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>
#include <string.h>

```

```

void revstr(char *str1){
    int i, len, temp;
    len = strlen(str1);
    for (i = 0; i < len/2; i++){
        temp = str1[i];
        str1[i] = str1[len - i - 1];
        str1[len - i - 1] = temp;
    }
}

```

```

int main(){

```

VISHAL DAS
2K21/A8/24

```
char str[50];  
printf ("Enter the string: ");  
gets(str);  
printf ("\nBefore reversing the string: %s \n", str);  
revstr(str);  
printf ("After reversing the string: %s\n", str);  
return 0;  
}
```

Output :



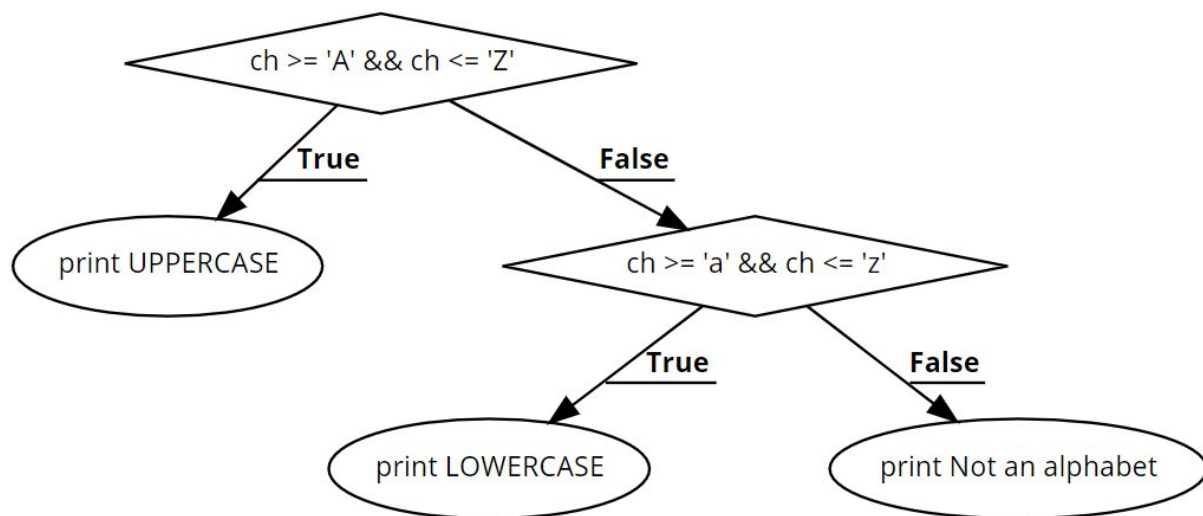
```
/home/vishal/Desktop/COE Lab/pr  
Enter the string: hello world  
  
Before reversing the string: hello world  
After reversing the string: dlrow olleh  
  
Process returned 0 (0x0)   execution time : 1.944 s  
Press ENTER to continue.  
█
```

Program 25 : Program to convert a string from lower case to upper case and vice versa

Theory :

The ASCII values of lowercase alphabets range from a=97 to z=122 and ASCII values of uppercase alphabets ranges from A=65 to Z=90. This means that the difference between any particular alphabet in its uppercase and lowercase format is 32. So, to convert uppercase to lowercase or vice versa, this difference of 32 is crucial.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 07, 2022 04:20 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>
#include <string.h>

int main(void){
    char string[100];
    int difference = 'A'-'a';
    printf("Input your string : ");
    fgets(string, sizeof(string), stdin);
    for(int i = 0 ; i < strlen(string); i++){
        if(string[i]>='A'&&string[i]<='Z')
            { string[i] -= difference; }
        else if(string[i]>='a'&&string[i]<='z')
            { string[i] += difference; }
    }
    printf("Case Changed : %s", string);
    return 0;
}

```



```
}
```

Output :

A terminal window with a title bar showing a gear icon, the path "/home/vishal/Desktop/COE L...", and three window control buttons (yellow, green, red). The terminal text is as follows:

```
Input your string : hElLo WorLd
Case Changed : HeLiO wORiD

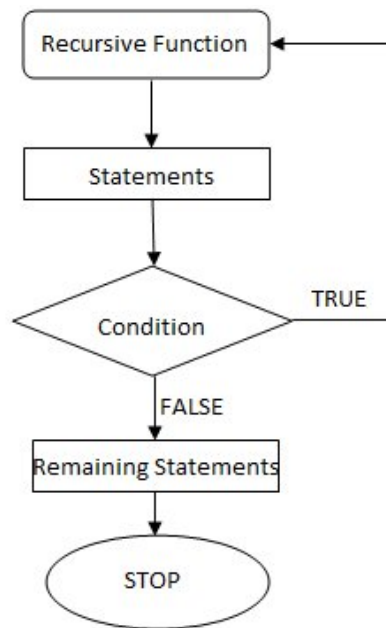
Process returned 0 (0x0)   execution time : 5.811 s
Press ENTER to continue.
█
```

Program 26 : Program to find factorial of a number using recursion

Theory :

A recursive function is a function that calls itself. The factorial function can be written as a recursive function call. Recall that $\text{factorial}(n) = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$. The factorial function can be rewritten recursively as $\text{factorial}(n) = n \times \text{factorial}(n - 1)$.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 14, 2022 03:40 PM
for Lab File of course C0102
*/
#include<stdio.h>

long int multiplyNumbers(int n) {
    if (n>=1)
        return n*multiplyNumbers(n-1);
    else
        return 1;
}

int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("\nFactorial of %d = %ld\n", n, multiplyNumbers(n));
    return 0;
}

```

VISHAL DAS
2K21/A8/24

Output :

A terminal window with a title bar showing a gear icon, the path "/home/vishal/Desktop/COE Lal", and standard window control buttons (yellow, green, red). The terminal text is as follows:

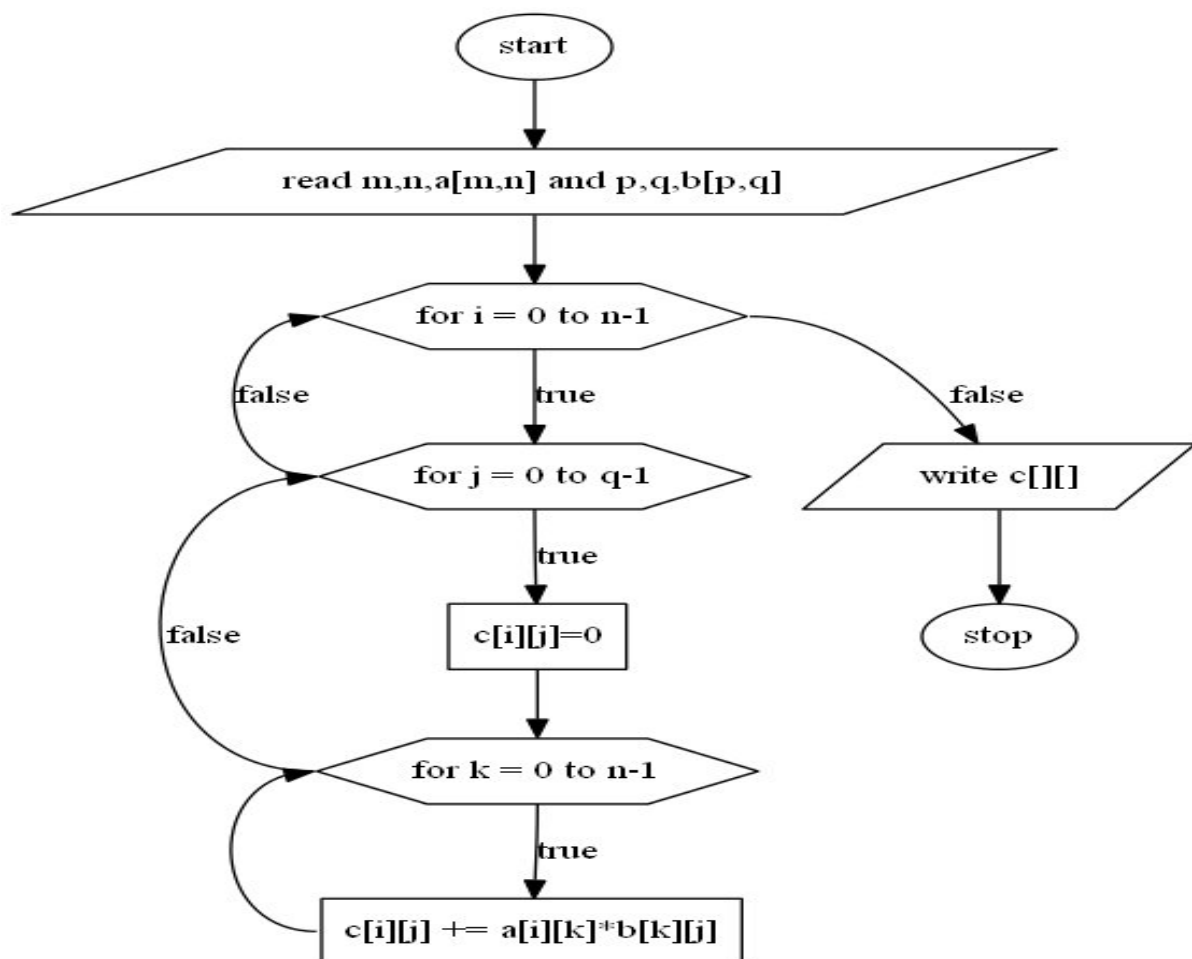
```
/home/vishal/Desktop/COE Lal
Enter a positive integer: 12
Factorial of 12 = 479001600
Process returned 0 (0x0)   execution time : 0,920 s
Press ENTER to continue.
█
```

Program 27 : Write a C program for the addition of two 3 x 3 matrices.

Theory :

In mathematics, a matrix (plural matrices) is a rectangular array or table of numbers, symbols, or expressions, arranged in rows and columns, which is used to represent a mathematical object or a property of such an object. If $A=[a_{ij}]$ and $B=[b_{ij}]$ are two matrices of the same order $m \times n$, then the sum of two matrix A and B is defined as $C=[c_{ij}]_{m \times n}$, where $c_{ij}=a_{ij}+b_{ij}$ for all possible values of i and j.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 14, 2022 04:10 PM
for Lab File of course C0102
*/
#include <stdio.h>

void add_mat(int mat1[3][3], int mat2[3][3], int result[3][3]){
    for (int i = 0; i < 3; i++){
        for (int j = 0; j < 3; j++){

```

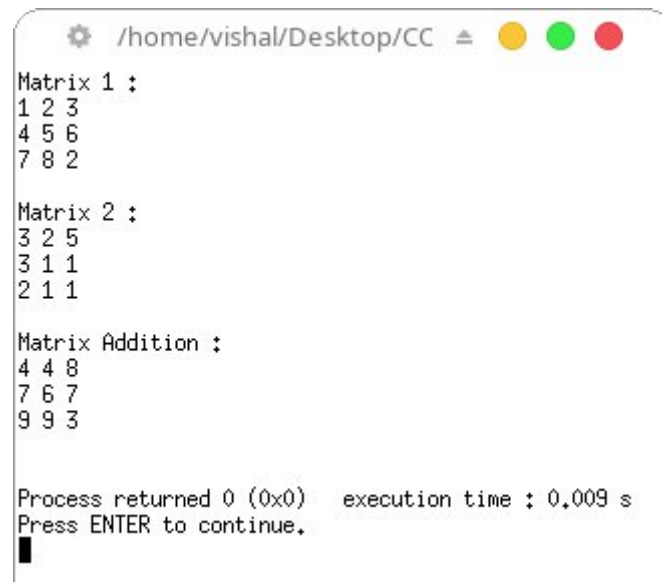
```

        result[i][j] = mat1[i][j] + mat2[i][j];
    }
}

int main(void)
{
    int mat1[3][3] = {{1,2,3}, {4,5,6}, {7,8,2}};
    int mat2[3][3] = {{3,2,5}, {3,1,1}, {2,1,1}};
    int out[3][3];
    add_mat(mat1, mat2, out);
    printf("Matrix 1 :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", mat1[i][0], mat1[i][1], mat1[i][2]);
    }
    printf("\nMatrix 2 :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", mat2[i][0], mat2[i][1], mat2[i][2]);
    }
    printf("\nMatrix Addition :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", out[i][0], out[i][1], out[i][2]);
    }
    printf("\n");
}

```

Output :



```

/home/vishal/Desktop/CC
Matrix 1 :
1 2 3
4 5 6
7 8 2

Matrix 2 :
3 2 5
3 1 1
2 1 1

Matrix Addition :
4 4 8
7 6 7
9 9 3

Process returned 0 (0x0)   execution time : 0.009 s
Press ENTER to continue.

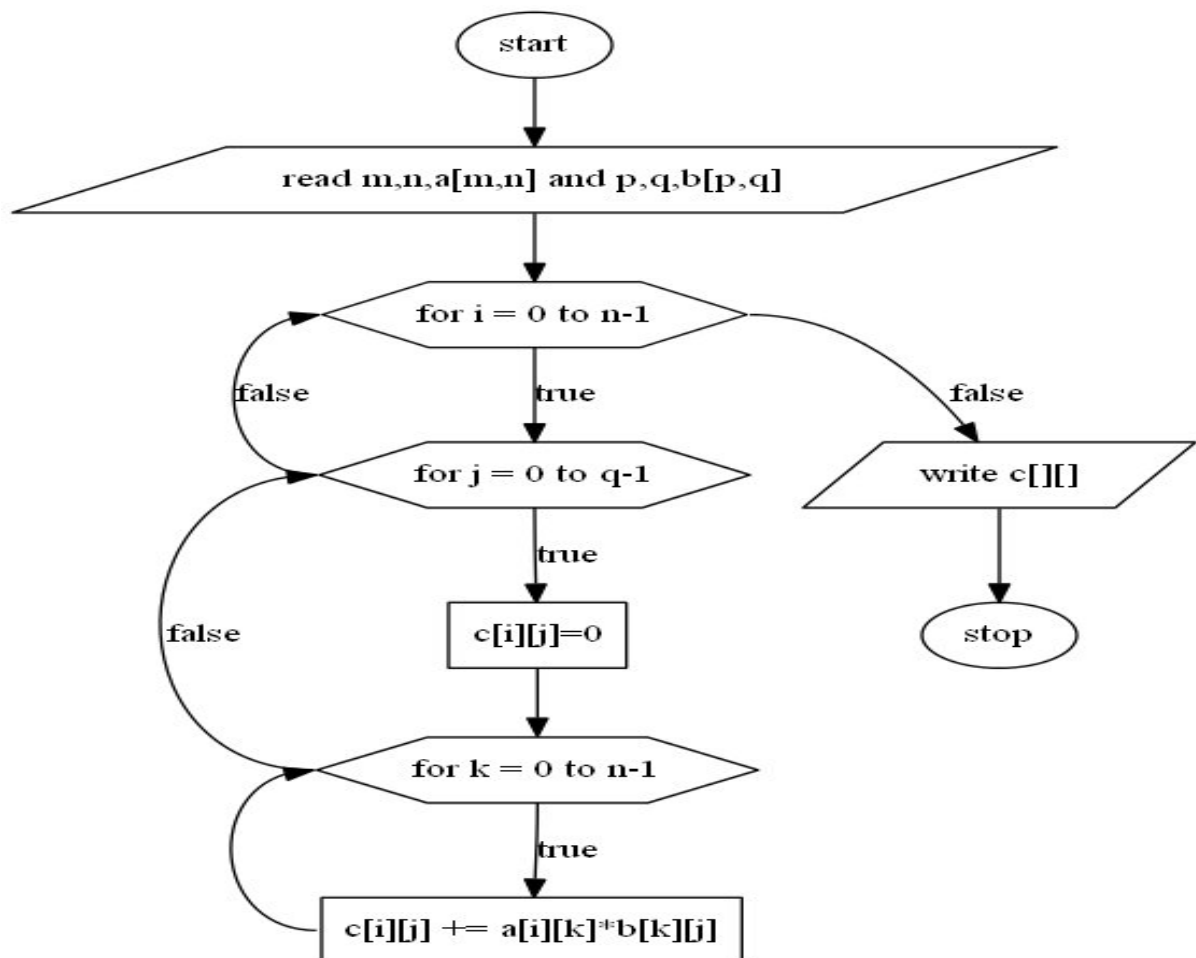
```

Program 28 : Write a C program to multiply two 3 x 3 matrices

Theory :

Matrix multiplication, also known as matrix product. The multiplication of two matrices, produces a single matrix. It is a type of binary operation. If A and B are the two matrices, then the product of the two matrices A and B are denoted by: $X = AB$. Hence, the product of two matrices is the dot product of the two matrices. To perform multiplication of two matrices, we should make sure that the number of columns in the 1st matrix is equal to the rows in the 2nd matrix. Therefore, the resulting matrix product will have a number of rows of the 1st matrix and a number of columns of the 2nd matrix.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 14, 2022 04:20 PM
for Lab File of course C0102
*/
#include <stdio.h>

void mul_mat(int mat1[3][3], int mat2[3][3], int result[3][3]){

```

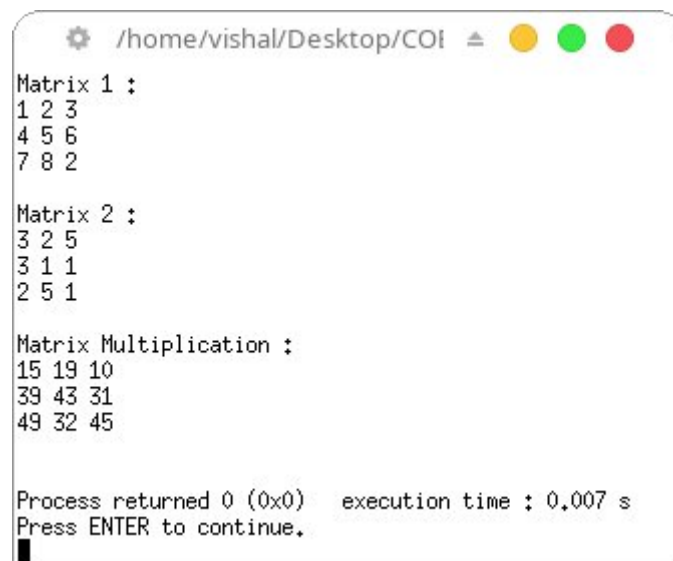
```

    for (int i = 0; i < 3; i++){
        for (int j = 0; j < 3; j++){
            int dot = mat1[i][0] * mat2[0][j] +
                      mat1[i][1] * mat2[1][j] +
                      mat1[i][2] * mat2[2][j];
            result[i][j] = dot;
        }
    }
}

int main(void)
{
    int mat1[3][3] = {{1,2,3}, {4,5,6}, {7,8,2}};
    int mat2[3][3] = {{3,2,5}, {3,1,1}, {2,5,1}};
    int out[3][3];
    mul_mat(mat1, mat2, out);
    printf("Matrix 1 :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", mat1[i][0], mat1[i][1], mat1[i][2]);
    }
    printf("\nMatrix 2 :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", mat2[i][0], mat2[i][1], mat2[i][2]);
    }
    printf("\nMatrix Multiplication :\n");
    for (int i=0; i < 3; i++){
        printf("%d %d %d\n", out[i][0], out[i][1], out[i][2]);
    }
    printf("\n");
}

```

Output :



```

/home/vishal/Desktop/COI
Matrix 1 :
1 2 3
4 5 6
7 8 2

Matrix 2 :
3 2 5
3 1 1
2 5 1

Matrix Multiplication :
15 19 10
39 43 31
49 32 45

Process returned 0 (0x0)   execution time : 0.007 s
Press ENTER to continue.

```

Program 29 : Write a C program to store the employee details using structure

Theory :

Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain many different data types (int, float, char, etc.).

Algorithm :

To make a struct in C, use the **struct** keyword inside the **main()** method, followed by the name of the structure and then the name of the structure variable:

```
struct myStructure {
    int myNum;
    char myLetter;
};
```

Code :

```
/* This program is written by Vishal Das
(2K21/A8/24) on July 14, 2022 04:50 PM
for Lab File of course C0102
*/
#include <stdio.h>
#include <stdlib.h>

typedef struct{
    char name[30];
    int id;
    double salary;
} Employee;

int main()
{
    int n=2;
    Employee employees[n];
    printf("Enter %d Employee Details \n \n",n);
    for(int i=0; i<n; i++){
        printf("Employee %d:- \n",i+1);
        printf("Name: ");
        scanf("%[^\n]s",employees[i].name);
        printf("Id: ");
        scanf("%d",&employees[i].id);
        printf("Salary: ");
        scanf("%lf",&employees[i].salary);
        char ch = getchar();
```



```

        printf("\n");
    }
    printf("----- All Employees Details ----- \n");
    for(int i=0; i<n; i++){
        printf("Name \t: ");
        printf("%s \n", employees[i].name);

        printf("Id \t: ");
        printf("%d \n", employees[i].id);

        printf("Salary \t: ");
        printf("%.2lf \n", employees[i].salary);

        printf("\n");
    }
    return 0;
}

```

Output :



```

/home/vishal/Desktop/COE L2
Employee 1:-
Name: Danti Tewari
Id: 05
Salary: 15000

Employee 2:-
Name: Yadav Sharma
Id: 14000
Salary: 1544

----- All Employees Details -----
Name      : Danti Tewari
Id         : 5
Salary    : 15000.00

Name      : Yadav Sharma
Id         : 14000
Salary    : 1544.00

Process returned 0 (0x0)   execution time : 18.903 s
Press ENTER to continue.

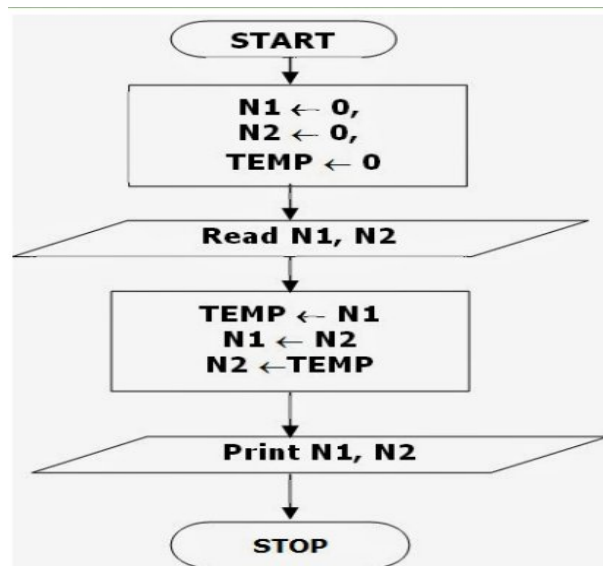
```

Program 30 : Write a C program to swap two numbers using pointers.

Theory :

The pointer will be used to hold the address of the variable and using pointer address of the variables will be swapped. The contents stored in the pointer variable are interchanged using a temporary variable.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 21, 2022 03:00 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>

```

```

void swap(int *x, int *y){
    int temp = *y;
    *y = *x;
    *x = temp;
}

```

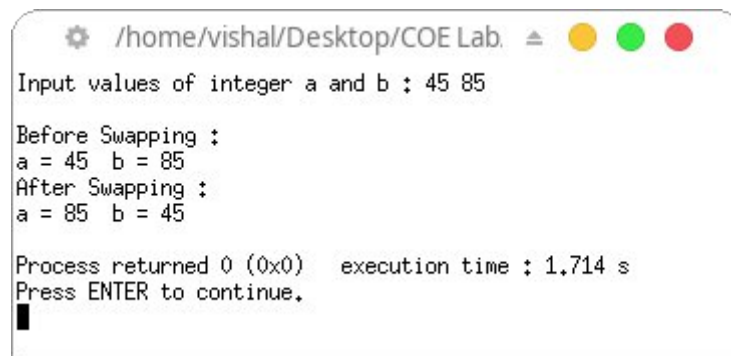
```

int main(void){
    int a, b;
    printf("Input values of integer a and b : ");
    scanf("%d %d", &a, &b);
    printf("\nBefore Swapping : \na = %d\tb = %d", a, b);
    swap(&a, &b);
    printf("\nAfter Swapping : \na = %d\tb = %d\n", a, b);
    return 0;
}

```

VISHAL DAS
2K21/A8/24

Output :

A terminal window with a title bar showing a gear icon, the path "/home/vishal/Desktop/COE Lab.", and three window control buttons (yellow, green, red). The terminal text is as follows:

```
/home/vishal/Desktop/COE Lab.
Input values of integer a and b : 45 85
Before Swapping :
a = 45  b = 85
After Swapping :
a = 85  b = 45
Process returned 0 (0x0)   execution time : 1.714 s
Press ENTER to continue.
█
```

Program 31 : Write a C program to pass and return pointer to function hence calculate average of an array.

Theory :

When an array in C language is declared, compiler allocates sufficient memory to contain all its elements. Its base address is also allocated by the compiler. If we declare an array arr, variable arr will give the base address, which is a constant pointer pointing to arr[0]. Hence arr contains the address of arr[0] i.e 1000. We use a pointer to an array, and then use that pointer to access the array elements.

Algorithm :

In this program, we have a pointer arr that points to the 0th element of the array. Similarly, we can also declare a pointer that can point to whole array instead of only one element of the array.

Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 21, 2022 03:15 PM
for Lab File of course C0102
*/

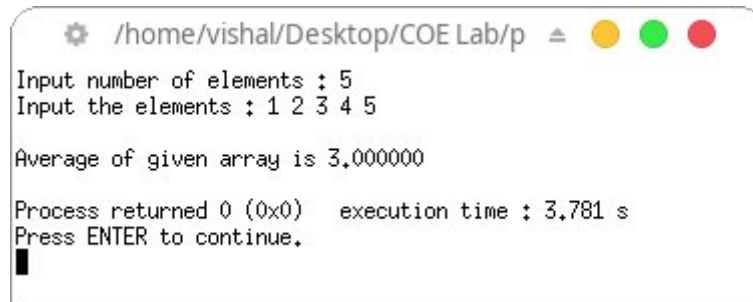
#include <stdio.h>

int average(int *array, int n){
    int i, sum=0;
    float avg;
    for(i=0;i<n;i++){
        sum += *(array+i);
    }
    avg=(float)sum/n;
    printf("\nAverage of given array is %f\n", avg);
    return avg;
}

int main(){
    int arr[100], n, i;
    printf("Input number of elements : ");
    scanf("%d",&n);
    printf("Input the elements : ");
    for(i=0;i<n;i++){
        scanf("%d", &arr[i]);
    }
    int *array = arr;
    average(array, n);
    return 0;
}

```

Output :

A terminal window with a title bar showing a gear icon, the path "/home/vishal/Desktop/COE Lab/p", and three window control buttons (yellow, green, red). The terminal text is as follows:

```
/home/vishal/Desktop/COE Lab/p
Input number of elements : 5
Input the elements : 1 2 3 4 5

Average of given array is 3.000000

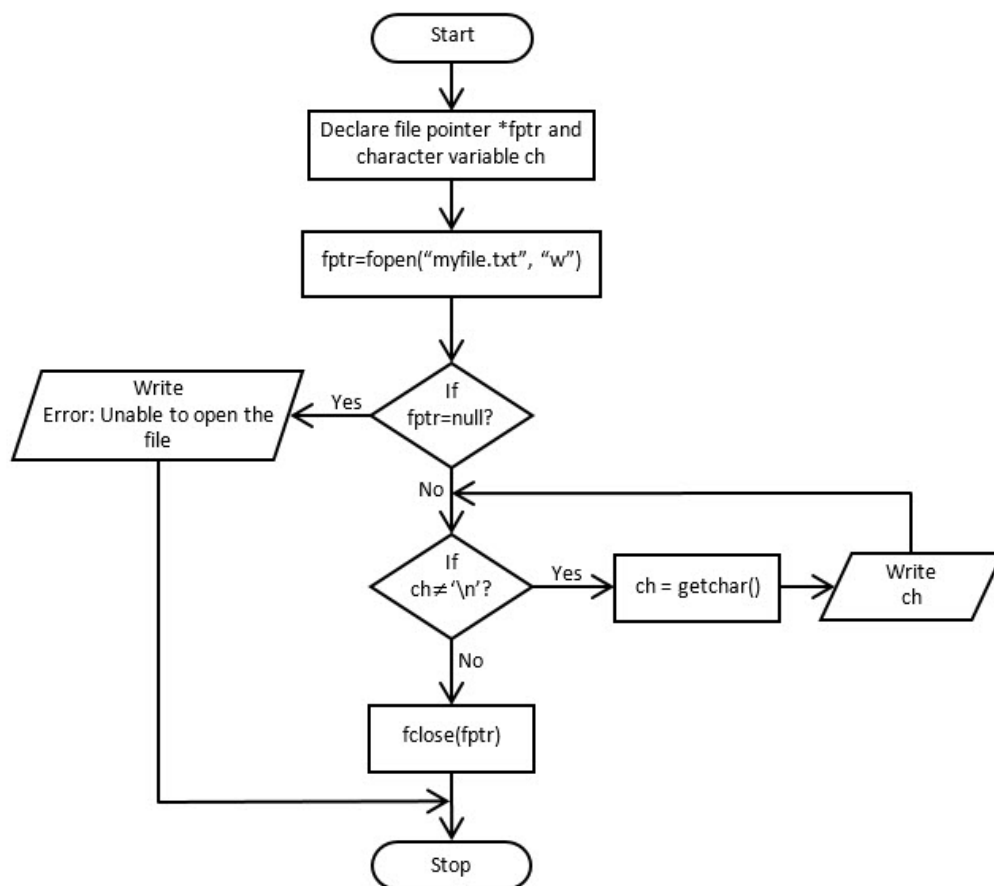
Process returned 0 (0x0)   execution time : 3.781 s
Press ENTER to continue.
█
```

Program 32 : Write a C program to create a file called *emp.txt* and store information about a person, in terms of his name, age and salary.

Theory :

The process of file handling enables a user to update, create, open, read, write, and ultimately delete the file/content in the file that exists on the C program's local file system. We open a file with the help of the `fopen()` function that is defined in the header file `stdio.h`.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 21, 2022 03:15 PM
for Lab File of course C0102
*/
#include <stdio.h>

int main(){
    char name[20];
    int age;
    float salary;

```

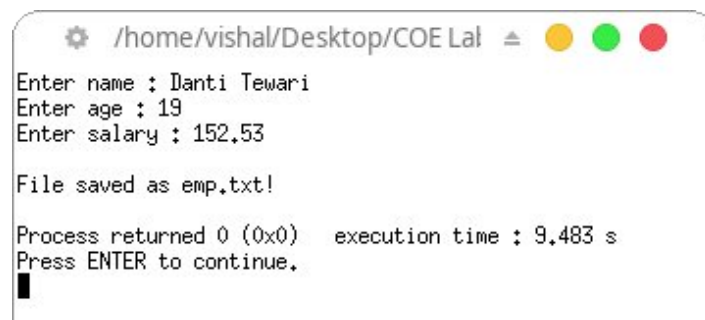
```

FILE *emp;
emp=fopen("emp.txt","w+");

printf("Enter name : ");
gets(name);
printf("Enter age : ");
scanf("%d",&age);
printf("Enter salary : ");
scanf("%f",&salary);
fprintf(emp,"%s %d %f",name,age,salary);
printf("\nFile saved as emp.txt!\n");
return 0;
}

```

Output :



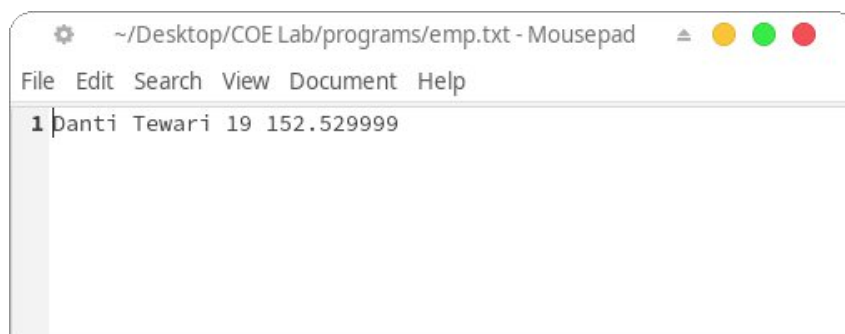
```

/home/vishal/Desktop/COE Lab
Enter name : Danti Tewari
Enter age : 19
Enter salary : 152.53

File saved as emp.txt!

Process returned 0 (0x0)   execution time : 9.483 s
Press ENTER to continue.

```



```

~/Desktop/COE Lab/programs/emp.txt - Mousepad
File Edit Search View Document Help
1 Danti Tewari 19 152.529999

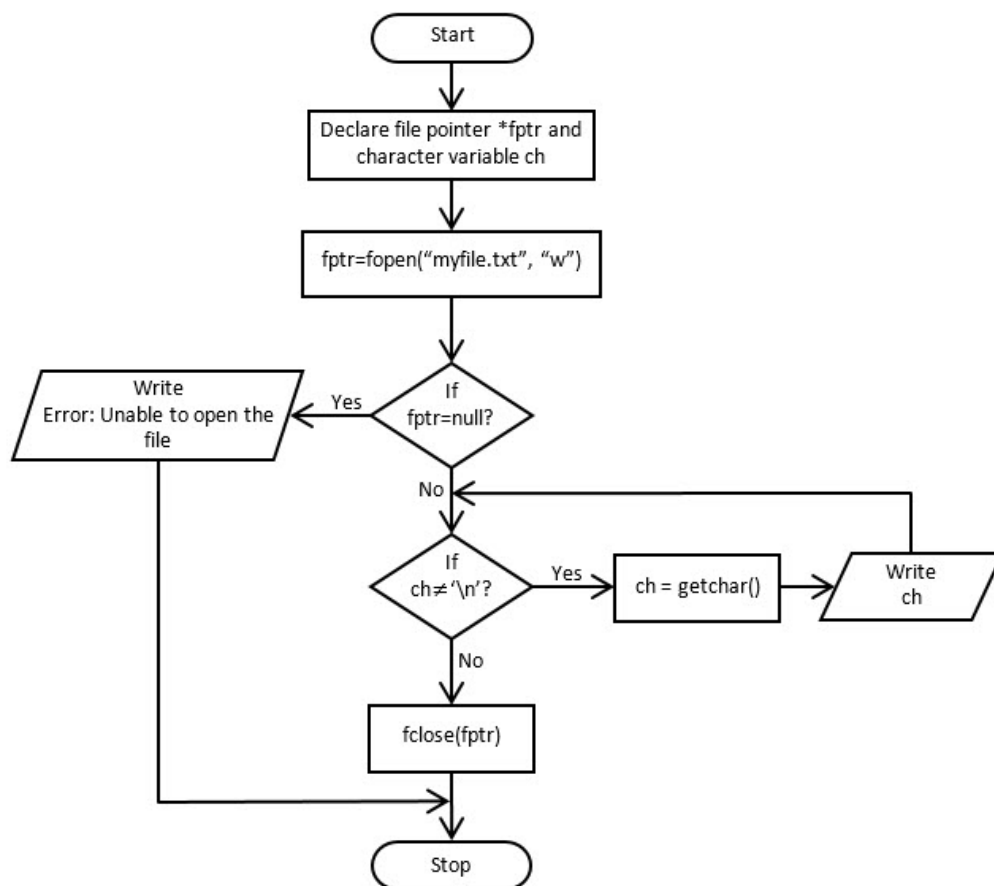
```

Program 33 : Write a C program to read a file and after converting all lower case to upper case letters write it to another file.

Theory :

The process of file handling enables a user to update, create, open, read, write, and ultimately delete the file/content in the file that exists on the C program's local file system. We open a file with the help of the `fopen()` function that is defined in the header file `stdio.h`.

Algorithm :



Code :

```

/* This program is written by Vishal Das
(2K21/A8/24) on July 21, 2022 03:15 PM
for Lab File of course C0102
*/

```

```

#include <stdio.h>
#include <string.h>

int main(){
    char text[100];
    FILE *emp, *new_file;

```

VISHAL DAS
2K21/A8/24

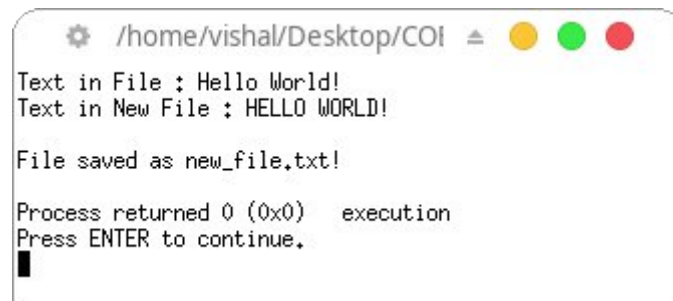

```

emp = fopen("text.txt","r");
new_file = fopen("new_file.txt", "w");
fgets(text, 100, emp);
printf("Text in File : %s\n", text);
for(int i=0;i<strlen(text);i++){
    if(text[i]>='a' && text[i]<='z')
        text[i] -= 32;
}
printf("Text in New File : %s\n", text);
fprintf(new_file,"%s", text);
printf("\nFile saved as new_file.txt!\n");

return 0;
}

```

Output :



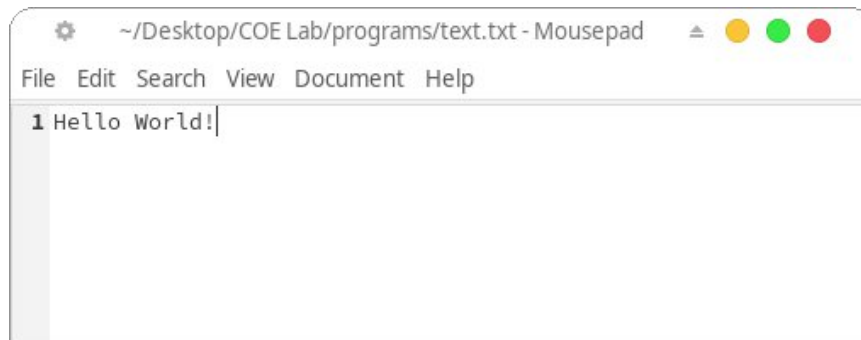
```

/home/vishal/Desktop/COI
Text in File : Hello World!
Text in New File : HELLO WORLD!

File saved as new_file.txt!

Process returned 0 (0x0)   execution
Press ENTER to continue.

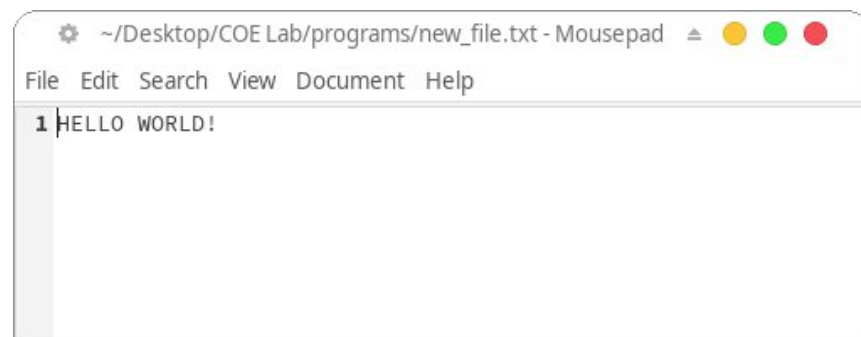
```



```

~/Desktop/COE Lab/programs/text.txt - Mousepad
File Edit Search View Document Help
1 Hello World!

```



```

~/Desktop/COE Lab/programs/new_file.txt - Mousepad
File Edit Search View Document Help
1 HELLO WORLD!

```