DELHI TECHNOLOGICAL UNIVERSITY

Programming Fundamentals Assignment

Submitted to:

Mr. Prashant G.

Submitted by:

Farhan Raja

Roll no. 2K19/A4/11

LIST OF EXPERIMENTS

Study topic: Study the working of a Compiler.

1. Program to find sum and average of two numbers.
2. Program to find greatest of 10 numbers.
3. Program to find Simple Interest.
4. Program to print the following pattern.(triangle of stars)
5. Program to fine whether the entered number is prime.
6. Program to find sum of a 5 digit number.
7. Program to reverse a 5 digit number.
8. Program to convert decimal to binary and vice versa.
9. Program to implement switch case statement.
10. Program to generate the Fibonacci sequence.
11. Program to find exponential function.
12. Program to search a number from an array using linear search.
13. Program to search a number from an array using binary search.
14. Program to sort an array using Bubble sort.
15. Program to sort an array using selection sort.
16. Program to sort an array using insertion sort.
17. Program to find factorial of a number using recursion.
18. Program to find the length of the string without using strlen and then pass the string to characters.
19. Program to count the number of vowels in a given string.
20. Program to check if a given string is a palindrome or not.
21. Program to string concatenation.
22. Program to string comparison.
23. Program to string reverse.
24. Program to convert a string from lower case to upper case and vice versa.
25. Program for the addition of two 3 x 3 matrices.
26. Program to multiply two 3 x 3 matrices
27. Program to swap two numbers using pointers.
28. Program to generate the employee details using structure.
29. Program to find the area and perimeter of a circle, rectangle, square and triangle using functions.
30. Program to pass and return pointer to function hence calculate average of an array.
31. Program to pass an array as pointer to a function that calculates the sum of all elements of the array.
32. Program to demonstrate the example of array of pointers.
33. Program to create a file called emp.txt and store information about a person, in terms of his name, age and salary.
34. Program which copies one file contents to another file.
35. Program to read a file and after converting all lower case to upper case letters write it to another file.
36. Program to find the size of a given file.

PROGRAM – 6

OBJECT:

Write a program to find sum of a five digit number.

INTRODUCTION:

The program accepts a value from the user and divides it with 10 and stores the remainder and then the number in its integer form is reduced by the factor of 10 and the process is repeated. This is carried out as long as the number reduced by the factor of 10 is greater 0 after reduction.

ALGORITHM:

Step-1: Start

Step-2: Declare the integer value n, sum and rem

Step-3: Take input from the user as n

Step-4: [Initialize] sum=0

Step-5: [Check Value] If n>0, then

[Compute] rem=n%10; sum=sum + rem and n=n/10

[Repeat loop] Go to Step-5

[End of loop]

Step-6: Print sum

Step-7: End

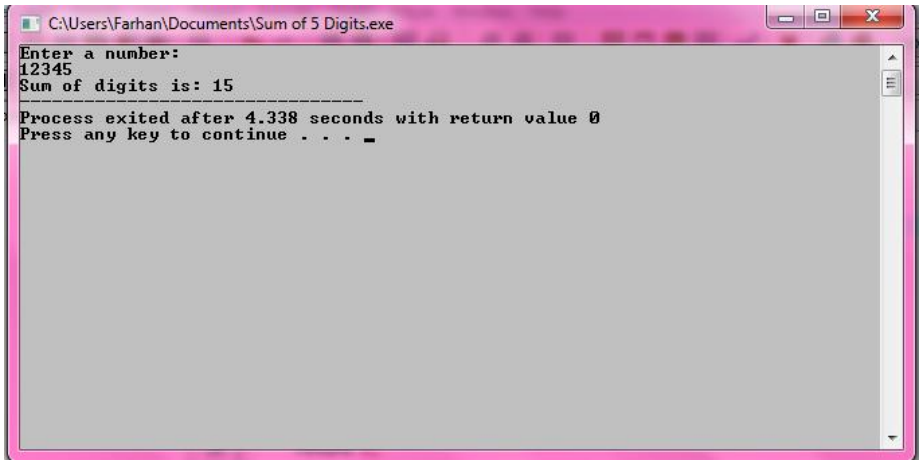
CODE:

//PROGRAM TO FIND SUM OF DIGITS OF A FIVE DIGIT NUMBER

```
#include<stdio.h>

int main()
```

```
{
        int n,rem,sum=0;
        printf("Enter a number:\n");
        scanf("%d",&n);
   while(n>0)
   {
         rem=n%10;
          sum=sum+rem;
          n=n/10;
     }
    printf("Sum of digits is: %d",sum);
    return 0;
}
```

Output:



LEARNING FROM EXPERIMENT:

We learned how to use while loop to read the digits of a number and to find its sum.

PROGRAM 7

AIM:

Write a program to reverse a five digit number.

INTRODUCTION:

This program reverse the number entered by the user and then prints the reversed number on the screen. For example if user enter 12345 as input then 54321 is printed as output. -In our program we use modulus(%) operator to obtain the digits of a number. To invert number look at it and write it from opposite direction or the output of code is a number obtained by writing original number from right to left.

ALGORITHM:

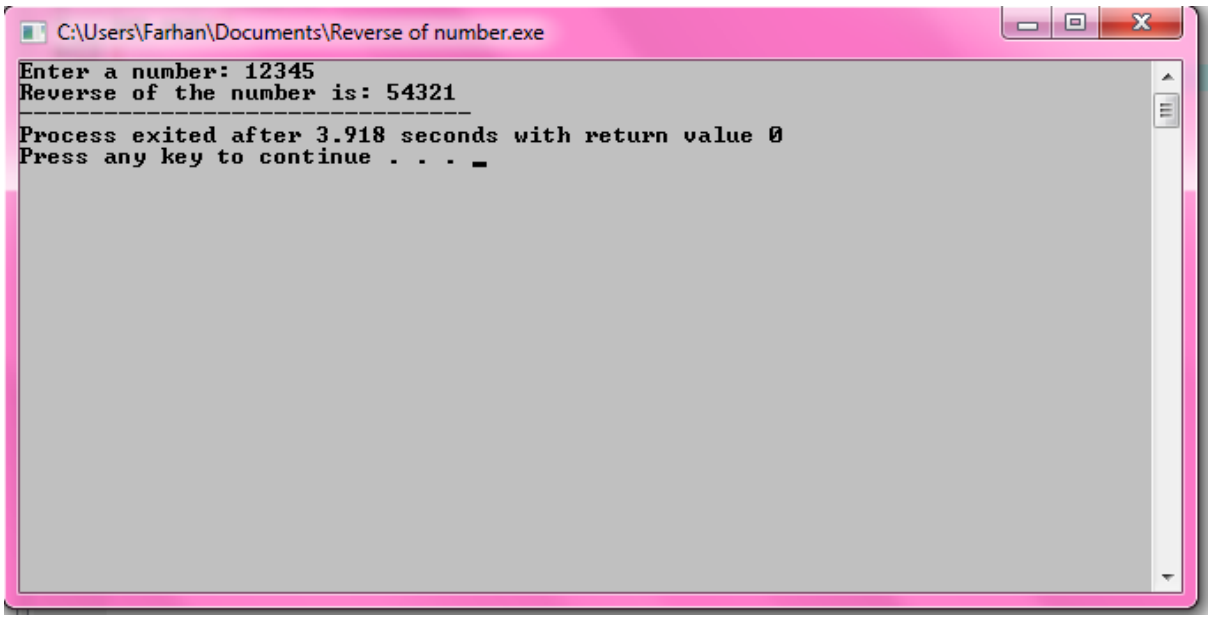
Step 1: Start
Step 2: Read: Take input for n
Step 3: [Initialize] reverse = 0
Step 4: [Check Value] If n > 0 then:
[Compute] rem = n%10
reverse = reverse*10 + rem and n = n/10]
[Repeat loop] Go to Step 4.
[End of If Structure]
Step 5: Print: reverse
Step 6: Exit

CODE:

```
//PROGRAMTO FIND REVERSE OF A NUMBER

#include<stdio.h>

int main()

{

        int n,rem,rev=0;

        printf("Enter a number: ");
```

```
        scanf("%d",&n);
        while(n>0)
        {
          rem=n%10;
          rev=rev*10+rem;
          n=n/10;
   }
   printf("Reverse of the number is: %d",rev);
   return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to use while loop for reversing a number.

PROGRAM 8

AIM:

Write a program to convert decimal to binary and vice versa.

INTRODUCTION:

Decimal to Binary conversion involves a number getting divided by 2 and the remainder noted, now the quotient of previous calculation is treated as a decimal and the process is repeated. As long as quotient is either 0 or 1, this process is repeated. The remainders are noted together and the reverse of this is the binary form of given decimal number.

Binary to Decimal conversion: The total number of terms in binary is noted, say n. To get decimal, the first term is multiplied by 2^{n-1}, second by 2^{n-2}, and so on till n=1. The sum of all these is taken and the resulting number is the decimal.

ALGORITHM:

1. Ask user whether he want to convert binary to decimal or decimal to binary
2. Create if else loop for conversion
3. To convert binary to decimal
   i) Ask user to enter a binary number and store it as binary
   ii) Create a while loop with condition binary!=0. Apply the following algorithm:

```
rem=binary%10;
dec=dec+rem*pow(2,i++)
binary=binary/10;
```

   iii) Display dec as output
4. To convert decimal to binary:
   i) Ask user to enter decimal number
   ii) Create a while loop with condition dec!=0 and apply the following algorithm:

```
rem=dec%2;
binary=binary+rem*pow(10,i++);
```

dec=dec/2;

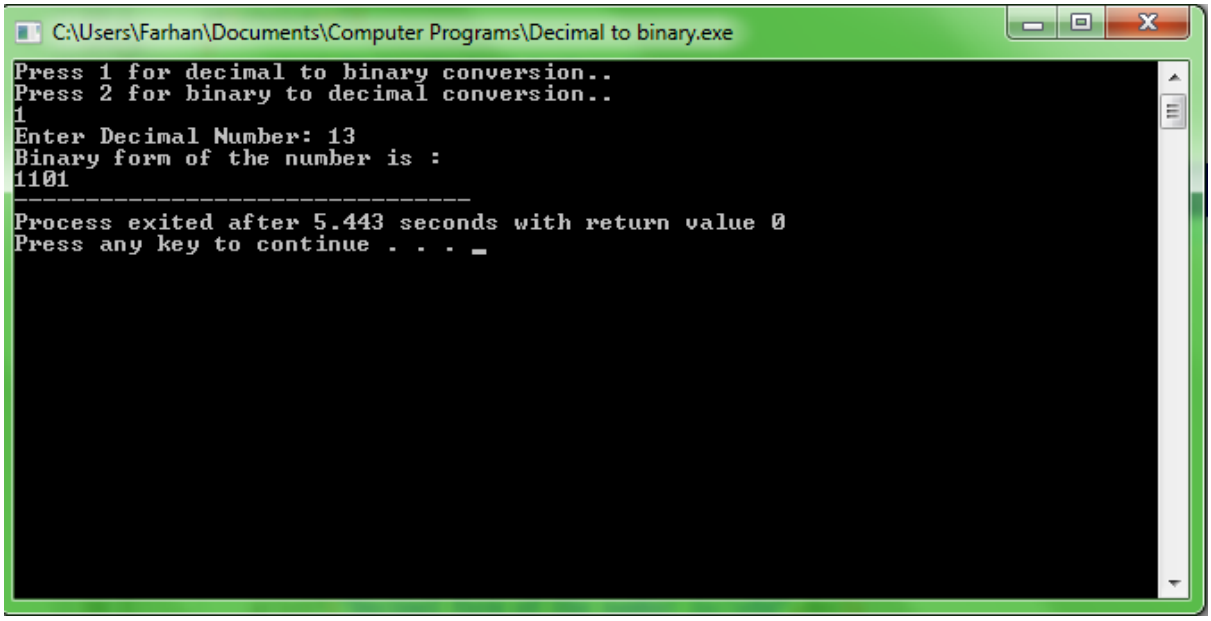
iii) Display binary as output

CODE:

```
//PROGRAM TO CONVERT DECIMAL TO BINARY AND VICE VERSA
#include<stdio.h>
#include<math.h>
#include<conio.h>
int main()
{
	int choice,dec=0,i=0,rem;
	long int binary=0;
	printf("Press 1 for decimal to binary conversion..\n");
	printf("Press 2 for binary to decimal conversion..\n");
	scanf("%d",&choice);
	if(choice==1)
	{
		printf("Enter Decimal Number: ");
		scanf("%d",&dec);
		while(dec!=0)
		{
			rem=dec%2;
			binary=binary+rem*pow(10,i++);
			dec=dec/2;
```

```
            }
            printf("Binary form of the number is :\n%d",binary);
        }
        if(choice==2)
        {
            printf("Enter Binary Number:\n");
            scanf("%d",&binary);
            while(binary!=0)
            {
                rem=binary%10;
                dec=dec+rem*pow(2,i++);
                binary=binary/10;
            }
            printf("Decimal form of the number is:\n%d",dec);
        }
        return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned the steps required for conversion of binary numbers to decimal numbers and vice versa.

Also the use of while loop for repetition and basic mathematical operations provided by C.

PROGRAM 9

AIM:

Write a program to implement switch case statement.

INTRODUCTION:

The switch() case in C programming allows case based programming. The user's response to test statement in the parentheses with 'switch' case gives the outcome associated with that response.

ALGORITHM:

Step1: Start

Step2: Declare variable n

Step3: Take input from user for n

Step4: Read input and execute the statement associated with that input

Step5: If no statement is associated with the given input, print "Out of range"

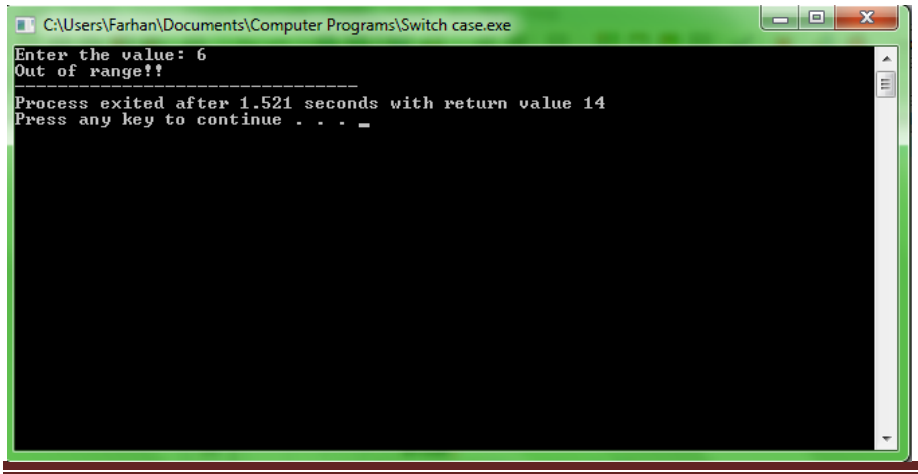
Step6: End

CODE:

```
#include<stdio.h>

void main()

{

        int n;

        printf("Enter the value: ");

        scanf("%d",&n);

        switch(n)
```

```
    {
        case 1: printf("You entered 1");
            break;
        case 2: printf("You entered 2");
            break;
        case 3: printf("You entered 3");
            break;
        case 4: printf("You entered 4");
            break;
        case 5: printf("You entered 5");
            break;
      default:
        printf("Out of range!!");
        break;
    }
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to use switch statements to make a menu driven program which can be used as an efficient replacement of if else nested loops for certain cases.

PROGRAM 10

AIM:

Write a program to generate Fibonacci sequence.

INTRODUCTION:

Fibonacci series is defined as a sequence of numbers in which the first two numbers are 1 and 1 and each subsequent number is the sum of the previous two. So, in this series, the nth term is the sum of (n-1)th term and (n-2)th term.

Mathematically, the nth term of the Fibonacci series can be represented as:

tn = tn-1 + tn-2

The Fibonacci sequence upto certain term can be represented as: 1, 1, 2, 3, 5, 8, 13, 21, 34…

ALGORITHM:

STEP 1. Start

STEP 2. Declare variables i, n , fib , t1, t2 .

STEP 3. Initialize the variables,t1=0, t2 =1, and fib=1 .

STEP 4. Enter the number of terms of Fibonacci series to be printed

STEP 5. Use loop for the following steps

Print fib

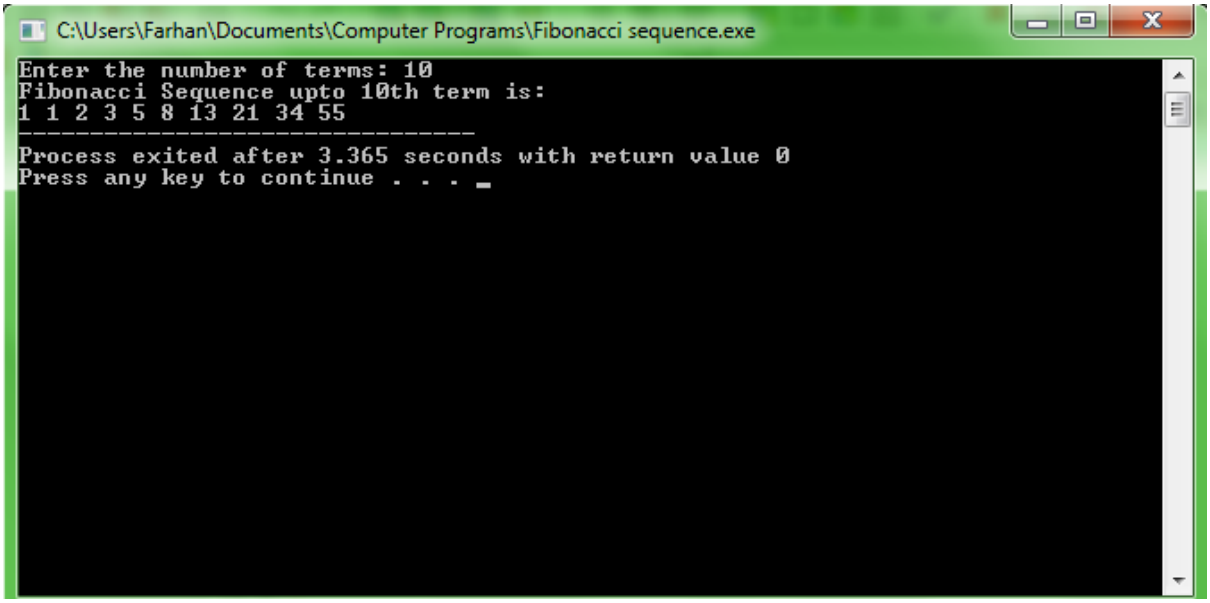
STEP 7. End

CODE:

```
//PROGRAM TO PRINT FIBONACCI SEQUENCE UPTO GIVEN NUMBER OF TERMS

#include<stdio.h>
```

```
int main()
{
	int i,n,fib=1,t1=0,t2=1;
	printf("Enter the number of terms: ");
	scanf("%d",&n);
	printf("Fibonacci Sequence upto %dth term is: \n",n);
	for(i=1;i<=n;i++)
	{
		printf("%d ",fib);
		fib=t1+t2;
		t1=t2;
		t2=fib;
	}
	return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned about Fibonacci Series. Experienced improvement in logic building skills.

PROGRAM 11:

AIM:

Write a program to find exponential function.

INTRODUCTION:

Exponential function is e^x. Value of exponential function can be calculated using Exponential Series.

The formula used to express the e^x as exponential series is

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Expanding the above notation, the formula of exponential series is

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

In this program user ask to find the exponential power of a value.

ALGORITHM:

Step1: Start

Step2: Declare integer variables i and n and float variables x t and sum

Step3: Initialize t=1 and sum =1

Step4: [Condition] for i=1 to i <=n

[Compute] t=t*x/I; and sum = sum + t; i++;

[Check the value of i]

[End loop]

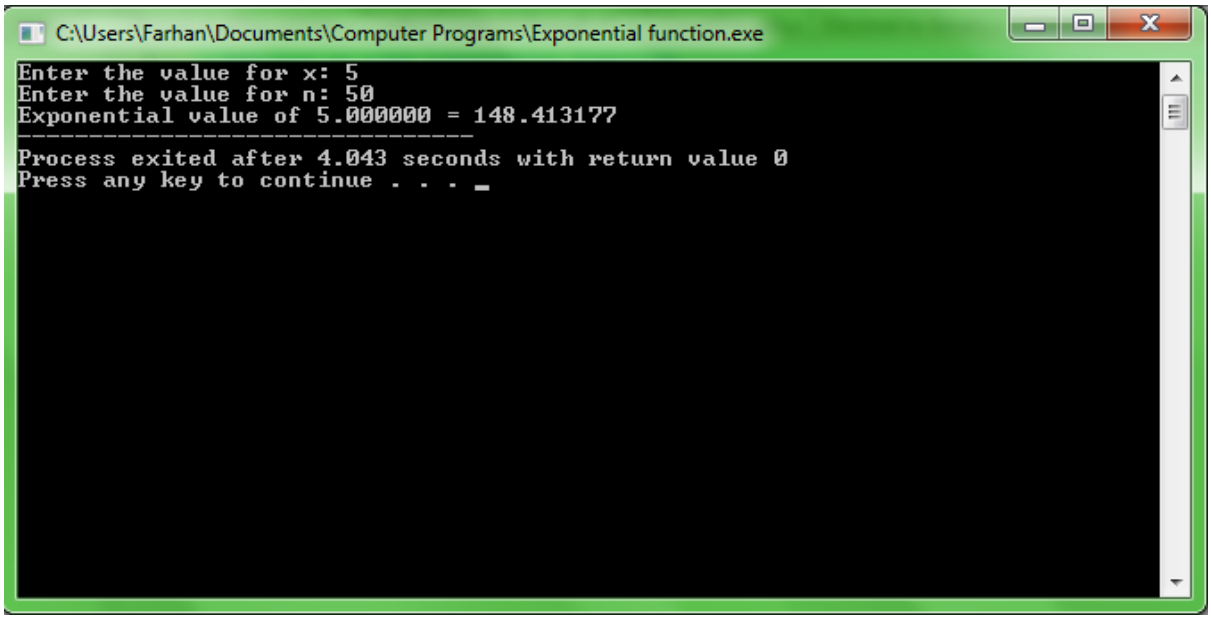
Step5: Print the value of sum

Step6: End

CODE:

```
//PROGRAM TO FIND EXPONENTIAL FUNCTION
#include<stdio.h>
#include<conio.h>
int main()
{
	int n,i;
	float x,t=1,sum=1;
	printf("Enter the value for x: ");
	scanf("%f",&x);
	printf("Enter the value for n: ");
	scanf("%d",&n);
	//Loop to calculate the value of Exponential
	for(i=1;i<=n;i++)
	{
		t=t*x/i;
		sum=sum+t;
	}
	printf("Exponential value of %f = %f",x,sum);
	return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned about exponential series and how to form series in programming language to calculate exact value of a function.

PROGRAM 12

AIM:

Write a program to search a number from an array using linear search.

INTRODUCTION:

Linear search or sequential search is a method for finding a particular value in a list, that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found.

ALGORITHM:

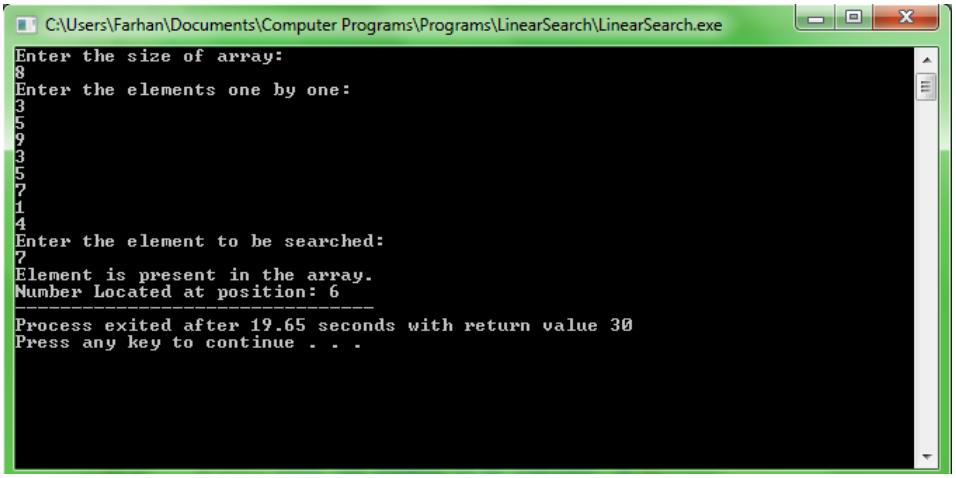
1. START
2. Declare an integer array and its size
3. Enter elements in array one by one
4. Enter the element to be found
5. Using for loop check each element in the array sequentially
6. If found, print 'Number is present' and its position in the array
7. END

CODE:

```
//PROGRAM TO FIND ELEMENTS OF ARRAY USING LINEAR SEARCH

#include <stdio.h>

void main()

{

        int array[10];

        int i, n, keynum, found = 0;



        printf("Enter the size of array: \n");

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

```
        printf("Enter the elements one by one: \n");
        for (i = 0; i < n; i++)
        {
                scanf("%d", &array[i]);
        }
        printf("Enter the element to be searched: \n");
        scanf("%d", &keynum);
        /* Linear search begins */
        for (i = 0; i < n ; i++)
        {
                if (keynum == array[i] )
                {
                        found = 1;
                        break;
                }
        }
        if (found == 1){
                printf("Element is present in the array.\n");
                printf("Number Located at position: %d ",i+1);
        }
        else
                printf("Element is not present in the array!\n");
}
```

OUTPUT:



LEARNING FROM THE EXPERIMENT:

We learned how to search elements of an array. We also learned how for loop can be used to read elements in an array one by one.

PROGRAM-13

OBJECT:

Program to search a number from an array using Binary search.

INTRODUCTION:

In computer science, **binary search**, also known as **half-interval search**, **logarithmic search**, or **binary chop**, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array; if they are unequal, the half in which the target cannot lie is eliminated and the search continues on the remaining half until it is successful or the remaining half is empty.

ALGORITHM:

1. START
2. Enter the size of array, say n
3. Declare an array, 'array' of given size
4. Begin for i=0 to n-1

   Input array[i], elements of the array

   End for
5. Sort the array using BubbleSort
6. Enter number to be searched
7. Compare the number with the middle element of the array(sub array)
8. If number is equal, go to step 11, if not go to step 9
9. If number is smaller than the middle value, consider half of the array containing smaller values and go to step 7
10. If number is greater than the middle value, consider half of the array containing larger values and go to step 7
11. Print the position of the number.
12. If not found print "SEARCH FAILED!"
13. END

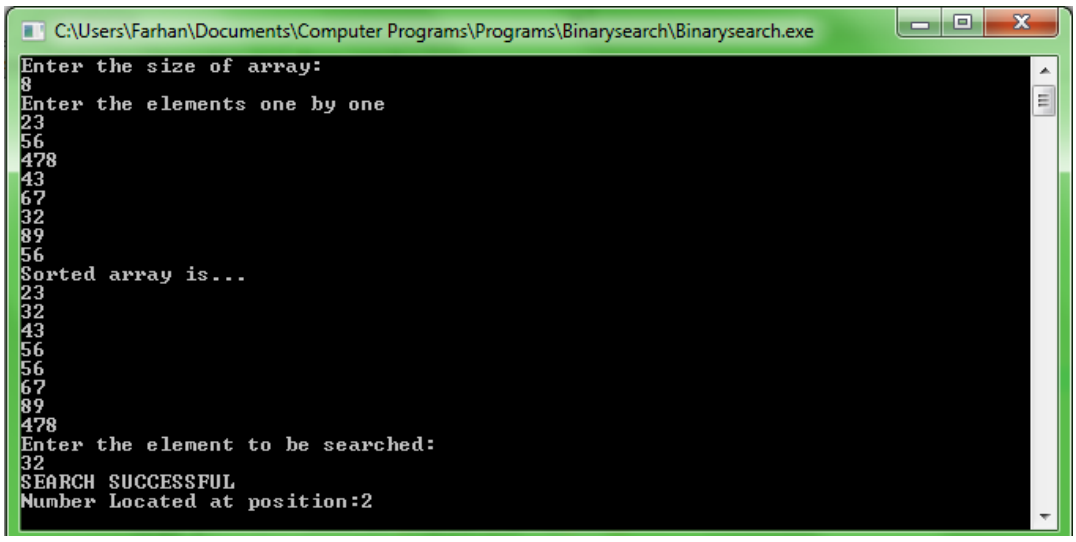
CODE:

```
#include <stdio.h>

void main()

{

        int array[10];

        int i, j, n, temp, keynum;

        int low, mid, high;



        printf("Enter the size of array: \n");

        scanf("%d", &n);



        printf("Enter the elements one by one \n");

        for (i = 0; i < n; i++)

        {

                scanf("%d", &array[i]);

        }



    /* Bubble sorting begins */

        for (i = 0; i < n; i++)

        {

                for (j = 0; j < (n - i - 1); j++)

                {

                        if (array[j] > array[j + 1])

                        {
```

```
                        temp = array[j];
                        array[j] = array[j + 1];
                        array[j + 1] = temp;
                }
        }
}
printf("Sorted array is...\n");
for (i = 0; i < n; i++)
{
        printf("%d\n", array[i]);
}
printf("Enter the element to be searched: \n");
scanf("%d", &keynum);
/*  Binary searching begins */
low = 1;
high = n;
do
{       mid = (low + high) / 2;
        if (keynum < array[mid])
                high = mid - 1;
        else if (keynum > array[mid])
                low = mid + 1;
} while (keynum != array[mid] && low <= high);
```

```
        if (keynum == array[mid])
        {
                printf("SEARCH SUCCESSFUL \n");
                printf("Number Located at position:%d ",mid+1);
        }
        else
        {
                printf("SEARCH FAILED! \n Number not found.");
        }
}
```

OUTPUT:



LEARNING FROM THE EXPERIMENT:

We learned about Binary search of elements in the array. We also learned how to Bubble sort array.

PROGRAM-14

OBJECT:

Program to sort array using Bubble sort.

INTRODUCTION:

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list.

ALGORITHM:

Step 1: START

Step 2: Repeat Steps 2 and 3 for i=1 to 10

Step 3: Set j=1

Step 4: Repeat while j<=n

(A) if array[i] < array[j]

Then interchange array[i] and array[j]

[End of if]

(B) Set j = j+1

[End of Inner Loop]

[End of Step 1 Outer Loop]

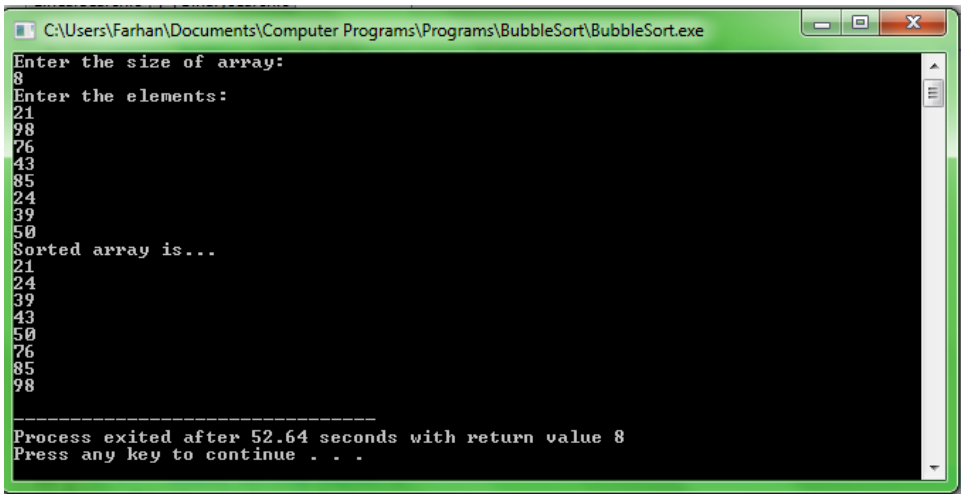
Step 5: END

CODE:

```
#include <stdio.h>

void main()
{
        int array[20];
        int i, j, n  , temp;
```

```
        printf("Enter the size of array: \n");
        scanf("%d", &n);

        printf("Enter the elements:\n");
        for (i = 0; i < n; i++)
        {
                scanf("%d", &array[i]);
        }
  /*  Bubble sorting begins */
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < (n - i - 1); j++)
                {
                        if (array[j] > array[j + 1])
                        {
                                temp = array[j];
                                array[j] = array[j + 1];
                                array[j + 1] = temp;
                        }
                }
        }

        printf("Sorted array is...\n");
        for (i = 0; i < n; i++)
        {
                printf("%d\n", array[i]);
        }
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to sort an array using a simple sorting method called Bubble sort.

PROGRAM-15

OBJECT:

Program to sort an array using Selection sort.

INTRODUCTION:

Selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains sub arrays in a given array.

ALGORITHM:

Let ARR is an array having N elements.

STEP 1: Start

STEP 2: Read arr

STEP 3: Repeat step 3 to 6 for i=0 to n-1

STEP 4: Set min=i

STEP 5: Repeat step 5 for j=i+1 to n

STEP 6: If arr[j]<arr[min], then

(a) Set min=j

[End of if]

[End of step 4 loop]

STEP 7: Interchange arr[i] and arr[min] using temporary variable

[End of step 2 outer loop]

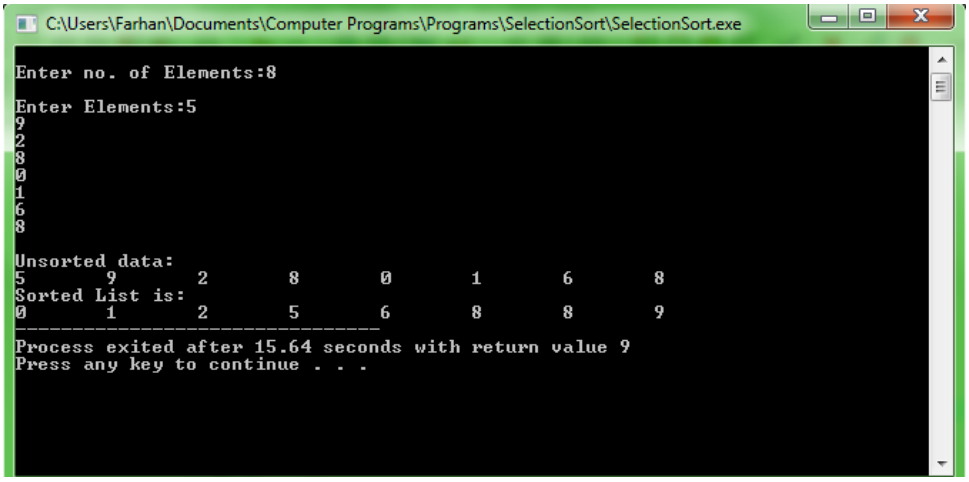
STEP 8: Stop

CODE:

```
//PROGRAM TO SORT ARRAY ELEMENTS USING SELECTION SORT
#include<conio.h>
#include<stdio.h>

void selsort(int, int[]);
void main()
{
	int i, size, arr[50];

	printf("\nEnter no. of Elements:");
	scanf("%d",&size);
	printf("\nEnter Elements:");
	for(i=1;i<=size;i++)
		scanf("%d",&arr[i]);

	printf("\nUnsorted data:\n");
	for(i=1;i<=size;i++)
		printf("%d\t",arr[i]);

	selsort(size, arr);
}
void selsort(int n, int arr[])
{	int i, j, min, temp;
	printf("\nSorted List is:\n");
	for(i=1;i<=n-1;i++)
	{
		min = i;
		for(j=i+1;j<=n;j++)
		{
			if(arr[j]<arr[min])
				min = j;
		}
		temp=arr[i];
		arr[i]=arr[min];
		arr[min]=temp;
	}
```

```
        for(i=1;i<=n;i++)
                printf("%d\t",arr[i]);
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to identify minimum value in array and to sort elements of array using selection sort.

PROGRAM-16:

OBJECT:

Program to sort an array using insertion sort:

INTRODUCTION:

This is an in-place comparison -based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there.

Hence the name, **insertion sort**.

ALGORITHM:

Step 1 – If it is the first element, it is already sorted. return 1;

Step 2 – Pick next element

Step 3 – Compare with all elements in the sorted sub-list

Step 4 – Shift all the elements in the sorted sub-list that is greater than the value to be sorted

Step 5 – Insert the value

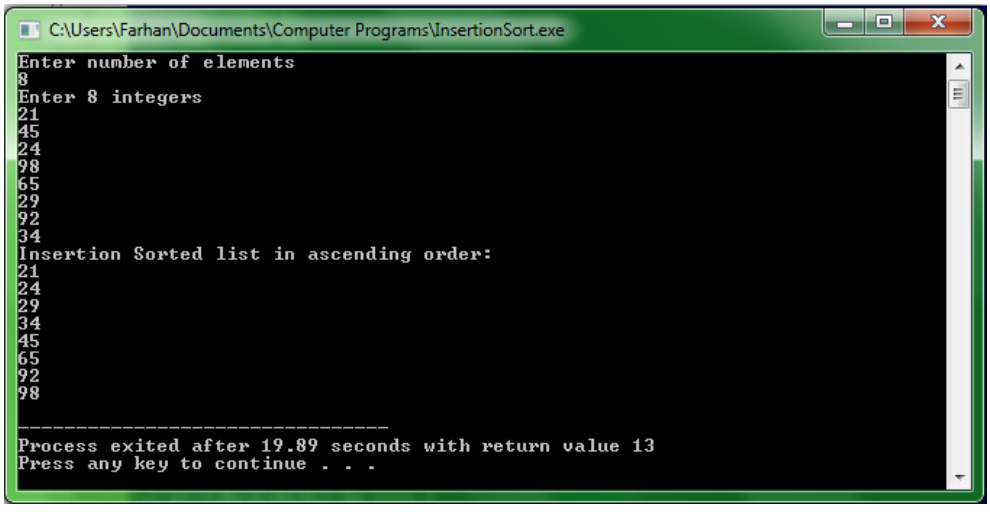
Step 6 – Repeat until list is sorted.

CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n, array[1000], c, d, t;

printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for (c = 0; c < n; c++)
```

```
{
   scanf("%d", &array[c]);
}
for (c = 1 ; c <= n - 1; c++)
{
     d = c;
     while ( d > 0 && array[d] < array[d-1])
     {
          t = array[d];
          array[d] = array[d-1];
          array[d-1] = t;
          d--;
   }
}
printf("Insertion Sorted list in ascending order:\n");
for (c = 0; c <= n - 1; c++)
 {
     printf("%d\n", array[c]);
 }
 getch();
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned to sort elements of array using Insertion sort.

PROGRAM-17

OBJECT:

Program to find factorial of a number using recursion.

INTRODUCTION:

The product of positive integers from 1 to N is called factorial N. It can also be defined as the factorial of a number N is defined as the product of first N Natural Numbers. Factorial N is denoted by N!

Where,

N! = 1.2.3.4.5……(N-2)(N-1).N

This function is defined for all positive integers including 0. Factorial of some positive integers are given below.

0! = 1

1! = 1

2! = 2.1 = 2

3! = 3.2.1 = 6

4! = 4.3.2.1 = 24

Factorial of fractions and negative numbers are not defined.

ALGORITHM:

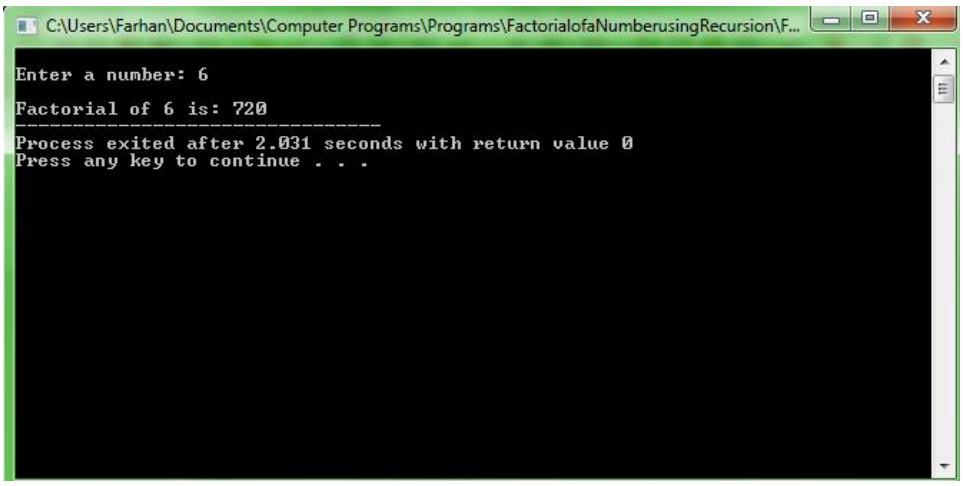
1. START
2. Declare a recursive function factorial()
3. Take input, n
4. If n=1 or n=0, return 1
   Else return n*factorial(n-1)
5. Print the value of n!
6. END

CODE:

```
//PROGRAM TO PRINT FACTORIAL OF A NUMBER USING RECURSION

#include<stdio.h>

int factorial(int);

int main(){

        int num,f;

        printf("\nEnter a number: ");

        scanf("%d",&num);


        f=factorial(num);


        printf("\nFactorial of %d is: %d",num,f);

        return 0;

}


//recursive function

int factorial(int n){

        if(n==1 || n==0)

                return 1;

        else
```

```
                return(n*factorial(n-1));

}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned about recursive functions and how to use them in programs to make it more portable.

PROGRAM-18

OBJECT:

Program to find length of the string without using strlen.

INTRODUCTION:

A string in C is merely an array of characters. The length of a string is determined by a terminating **null character**: '\0'. So, a string with the contents, say, "abc" has four characters: 'a', 'b', 'c', and the terminating **null character**. The terminating **null character** has the value zero.

Length of String

This operation is used to count number of characters in the string.

In order to count number of characters, we have to start counting from the first character in the string which is at index 0 in the array. The process of counting characters is repeated until NULL character is encountered in the string, because NULL character indicates end of the string.

ALGORITHM:

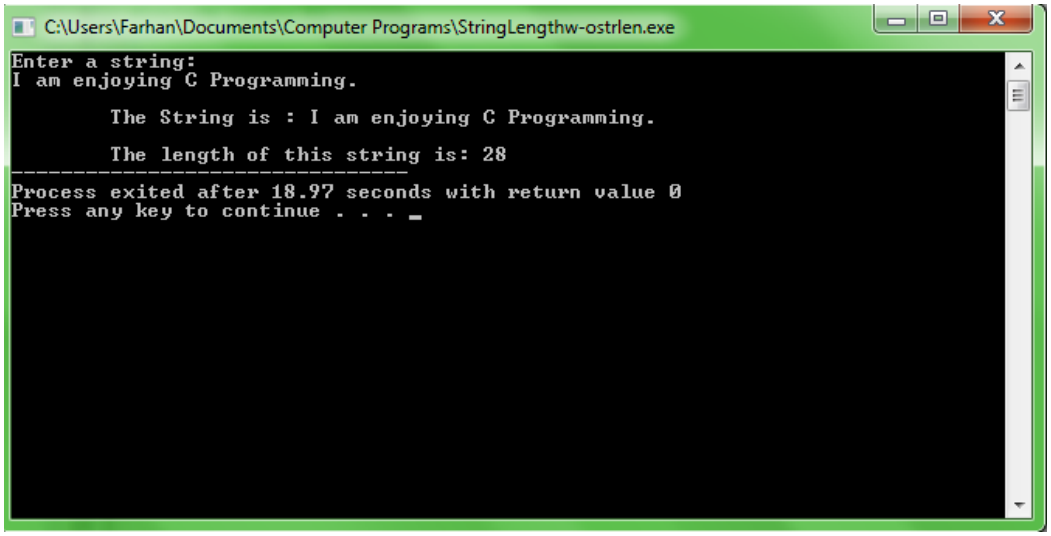
1. Declare a function string_function() to pass string to a function
2. Take a string as input and store it in the array string[].
3. Using for loop count the number of characters in the array string[]. Use the variable length to keep the count of the characters in the array.
4. Do step-2 till the end of input string.
5. Print the variable length as output

CODE:

```
#include <stdio.h>

/*

Program to pass a string to a function

*/
```

```
void string_function(char *ptr)
{
    printf("\n\tThe String is : %s",ptr);
}

// main function
int main()
{
        int i,length=0;
        char string[100];

        printf("Enter a string:\n");
        gets(string);
        //Finding length of string
        for(i=0; string[i] != '\0'; i++)
        {
                length++;
        }
        string_function(string);
        printf("\n\n\tThe length of this string is: %d",length);
        return 0;
}
```

OUTPUT:



LEARNING FROM THE EXPERIMENT:

We learned how to pass string to a function and we also learned how to find length of a sting using for loop.

PROGRAM-19

OBJECT:

Program to count the number of vowels in a given string.

INTRODUCTION:

C program to count number of vowels in a string: for example, in the string "love" there are two vowels 'o' and 'e'. In the program both lower and upper case are considered i.e., 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u' and 'U'. In this program we check every character in the input string, if it's a vowel then counter is incremented by one, consonants and special characters are ignored.

ALGORITHM:

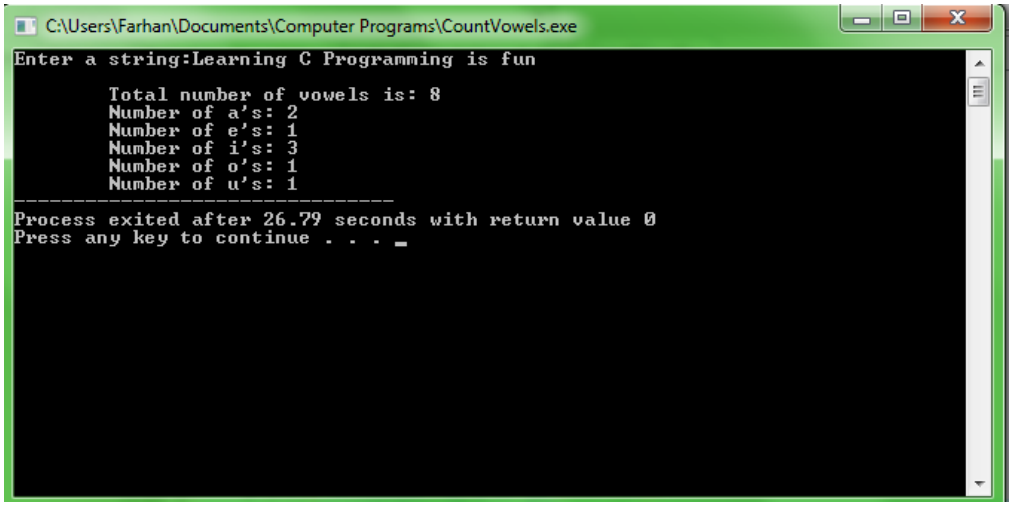
1. Take string as input from the user
2. Read string one by one using while loop till string[i] != '\0'
3. If string[i] is a vowel increase the counter by one
4. End of loop
5. Print the total number of vowels.

CODE:

```
/*Programm to count the number of vowels in a given string*/

#include<stdio.h>

int main()

{

   int i=0,sum, count_a=0, count_e=0, count_i=0, count_o=0, count_u=0;

         char string[100];

         printf("Enter a string:");

         gets(string);
```

```
        while(string[i]!='\0')
        {
                if(string[i] == 'a'||string[i]=='A')
                { count_a++;}
                if(string[i] == 'e'||string[i]=='E')
                {count_e++;}
                if(string[i] == 'i'||string[i]=='I')
                {count_i++;}
                if(string[i] == 'o'||string[i]=='O')
                {count_o++;}
                if(string[i] == 'u'||string[i]=='U')
                {count_u++;}
        i++;
        }
        sum = count_a + count_e + count_i + count_o + count_u;
        printf("\n\tTotal number of vowels is: %d", sum);
        printf("\n\tNumber of a's: %d",count_a);
        printf("\n\tNumber of e's: %d",count_e);
        printf("\n\tNumber of i's: %d",count_i);
        printf("\n\tNumber of o's: %d",count_o);
        printf("\n\tNumber of u's: %d",count_u);
   return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to count certain characters in a string and also how to use basic logical operators in a program.

PROGRAM-20

OBJECT:

Program to check if a given string is palindrome or not.

INTRODUCTION:

The program is used to check a given string is palindrome or not.

Palindromes are words or phrases that read the same in both directions.

ALGORITHM:

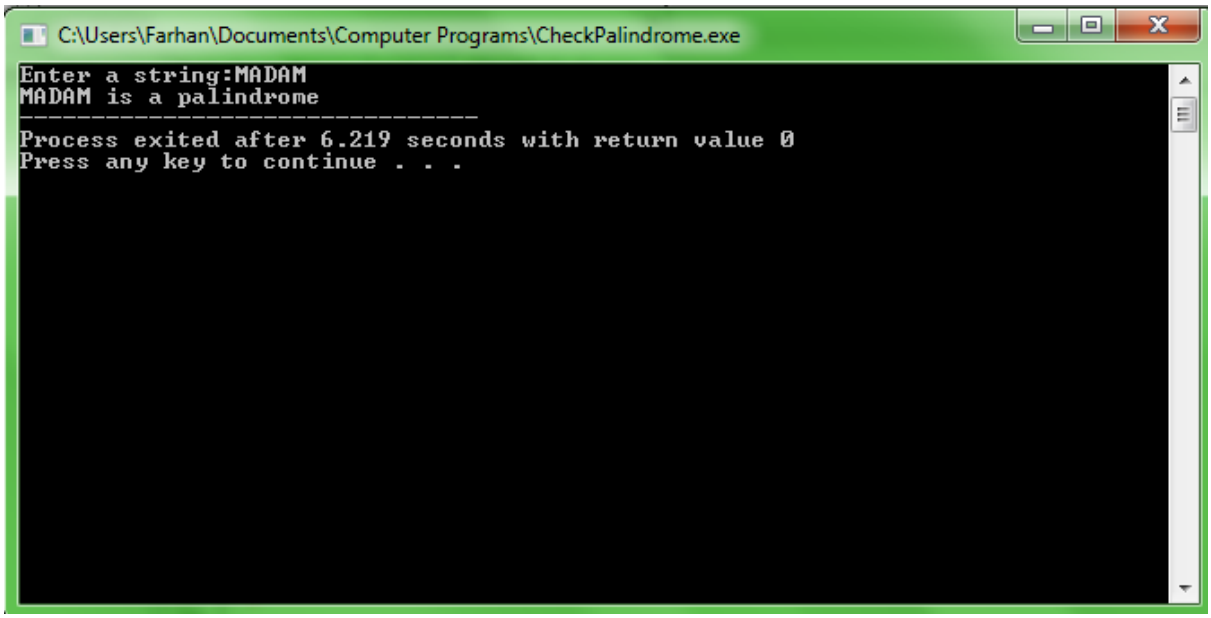
1. The string is asked from the user and stored it in a string
2. The length of the string is stored in length using strlen().
3. In iteration if starting letters != last letters then flag=1.
4. If flag=1, print not palindrome otherwise palindrome.

CODE:

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

int main(){
   char string[50];
   int i, length;
   int flag = 0;

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

   length = strlen(string);

   for(i=0;i < length ;i++){
      if(string[i] != string[length-i-1]){
         flag = 1;
         break;
  }
```

```
}

   if (flag) {
      printf("%s is not a palindrome", string);
   }
   else {
      printf("%s is a palindrome", string);
   }
   return 0;
}
```

OUTPUT:



LEARNING FROM THE EXPERIMENT:

We learned about palindromes. We also learned about strlen function and how to iterate to check given word is palindrome or not.

PROGRAM-21

OBJECT:

Program to string concatenation.

INTRODUCTION:

In formal language theory and computer programming, **string concatenation** is the operation of joining character strings end-to-end. For example, the concatenation of "snow" and "ball" is "snowball".

ALGORITHM:

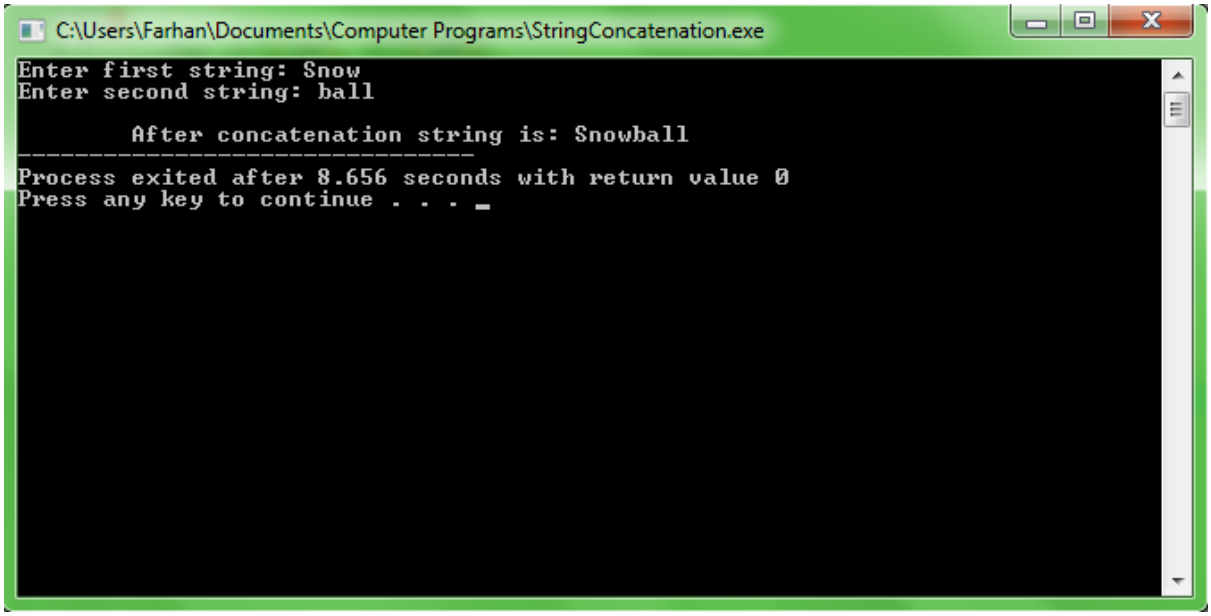
STEP1: Initialize two character arrays str1, str2
STEP2: Input two stings from user
STEP3: Traverse in str1 character by character
Increment i by 1 till we reach null character
STEP4: Add srt2 at the end of str1 i.e. from i character by character
STEP5: Add a null character at the end of new str1 string
STEP6: End

CODE:

```
//PROGRAM TO CONCATENATE TWO STRINGS

#include<stdio.h>

int main()

{

        char str1[100], str2[100];

        printf("Enter first string: ");

        gets(str1);

        printf("Enter second string: ");

        gets(str2);
```

```
        int i,j;
   for(i = 0; str1[i] != '\0'; ++i);
      for(j = 0; str2[j] != '\0'; ++j, ++i)
      str1[i] = str2[j];
      str1[i] = '\0';
   printf("\n\tAfter concatenation string is: %s",str1);
        return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMEN:

We learned string properties and how to concatenate two strings without using strcat(), but using for loop.

PROGRAM-22

OBJECT:

Program to string comparison.

INTRODUCTION:

Comparing strings is the process of analysing and outputting the differences or similarities between two strings. Comparing strings has many uses like checking differences or similarities in a user response. Two variables may typically have differences in characters. For example, of variable 1 has a value of "potato" and variable 2 has a value of "tomato" the differences would be "pm" because those are the characters not shared by both variables. Only variable 1 contains a "p" and only variable 2 contains an "m".

This program prompts the user to enter two strings and check whether the two strings are same or have same length or not using a for loop.

ALGORITHM:

1. START
2. Declare two char arrays named string1 and string2 to store the strings entered by the user.
3. Input 2 strings
4. Find length of the two strings. If they are same print same length and go to step 5 if not print strings are not same, end program
5. Using a for loop form i=0 to i<length check if every character of the two string is same or not if same print same if not print not same
6. END

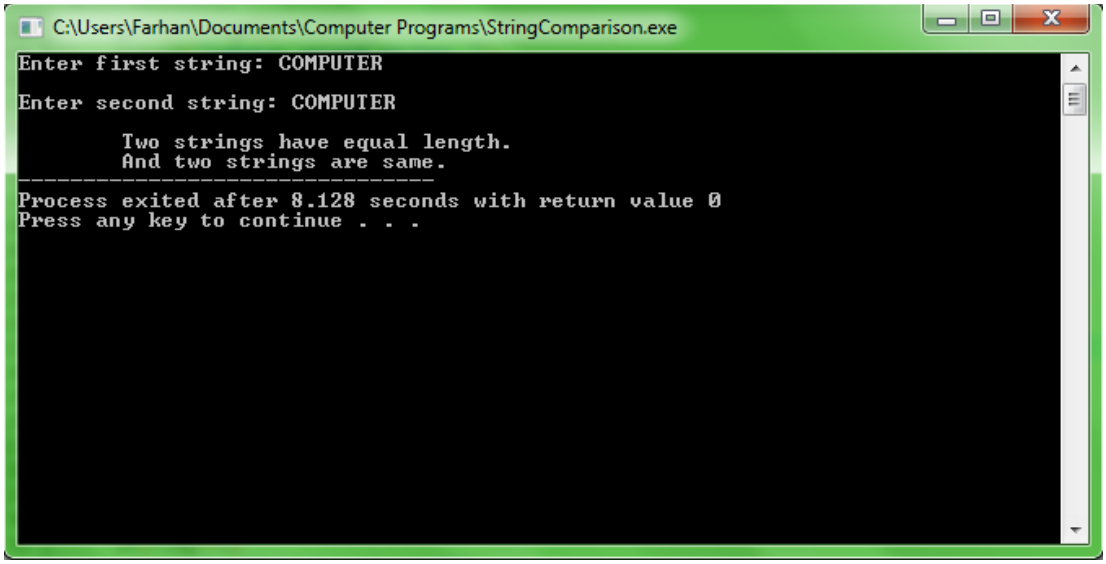
CODE:

```
#include<stdio.h>

#include<string.h>
```

```
#include<stdlib.h>

int main()

{

        char string1[100], string2[100];

        int i,x,y, flag=0;


        printf("Enter first string: ");

        gets(string1);

        printf("\nEnter second string: ");

        gets(string2);


        x=strlen(string1);

        y=strlen(string2);

        if(x!=y)

        {

           printf("\n\tTwo strings are not same!!");

    exit(0);

        }

        else

        {

          printf("\n\tTwo strings have equal length.");

  }

        for(i=0; string1[i]; i++)
```

```
        {
                if(string1[i] != string2[i])
                flag=1;
        }
        if(flag==1)
        printf("\n\tBut are not same!!");
        else
        printf("\n\tAnd two strings are same.");
        return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to calculate the length of string using strlen function.

We also learned how to compare to strings and check if they are same or not.

We also learned about the use of exit function.

PROGRAM-23

OBJECT:

Program to string reverse.

INTRODUCTION:

A string in C is merely an array of characters. The length of a string is determined by a terminating **null character**: '\0' . So, a string with the contents, say, "abc" has four characters: 'a' , 'b' , 'c' , and the terminating **null character**. The terminating **null character** has the value zero.

Reverse of string
First we calculate the length of string using strlen function and then assigns characters of input string in reverse order to create a new string using a for loop.
So, this program reverses a string entered by the user. For example if a user enters a string "reverse me" then on reversing the string will be "em esrever".

ALGORITHM:

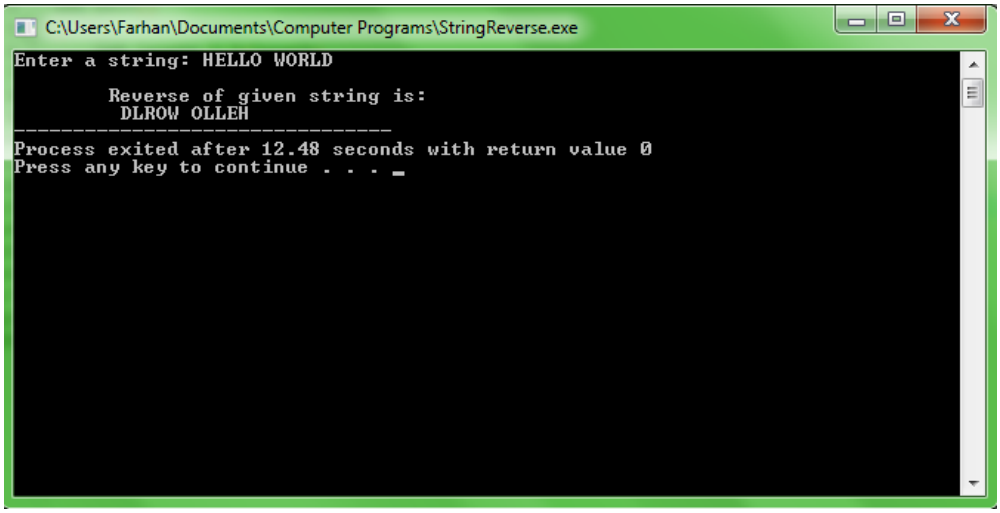
Step 1: Start
Step 2: Length = 0
Step 3: Repeat step 4 while string[length] ≠ NULL
Step 4: length = length + 1
Step 5: Return length
Step 6: Take 2 Subscript Variables i, j
Step 7: j is positioned on **Last Character**
Step 8: i is positioned on **first character**
Step 9: string[i] is **interchanged** with string[j]
Step 10: Increment i
Step 11: Decrement j
Step 12: If i > j then goto step 3
Step 13: Stop

CODE:

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

```
#include<string.h>


int main()

{

        int length,i,j=length;

        char string[100];

        printf("Enter a string: ");

        gets(string);


        length = strlen(string);


        printf("\n\tReverse of given string is:\n\t");

        for(i=length; i>=0; i--)

        {

          printf("%c",string[i]);

   }

        return 0;

}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learn how to effectively reverse a string using a for loop.

PROGRAM–24

OBJECT:

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

INTRODUCTION:

This program converts the case of each letter that is upper to lower and vice versa.

What is the upper case: letters A – Z are upper.
What is lowercase: letters a – z is lower.
In c , the ASCII value of letters :
A = 65 , B = 66 , ….. , Z = 90.
a = 97 , b = 98 , …… , z = 122.
We can use the ASCII values of the letter based on that we convert upper case to lower case and vice versa.

To convert upper to lower: add 32 with an uppercase letter.
Example:
convert A to a
=> A + 32
=> 65 + 32
=> 97
And 97 is ASCII value of a

The logic to convert lower to upper: subtract 32 with lower case letter.
Example:
convert a to A
=> a – 32
=> 97 – 32
=> 65
And 65 is ASCII value of A

ALGORITHM:

1. Take the input string from the user.
2. Find the length of the string

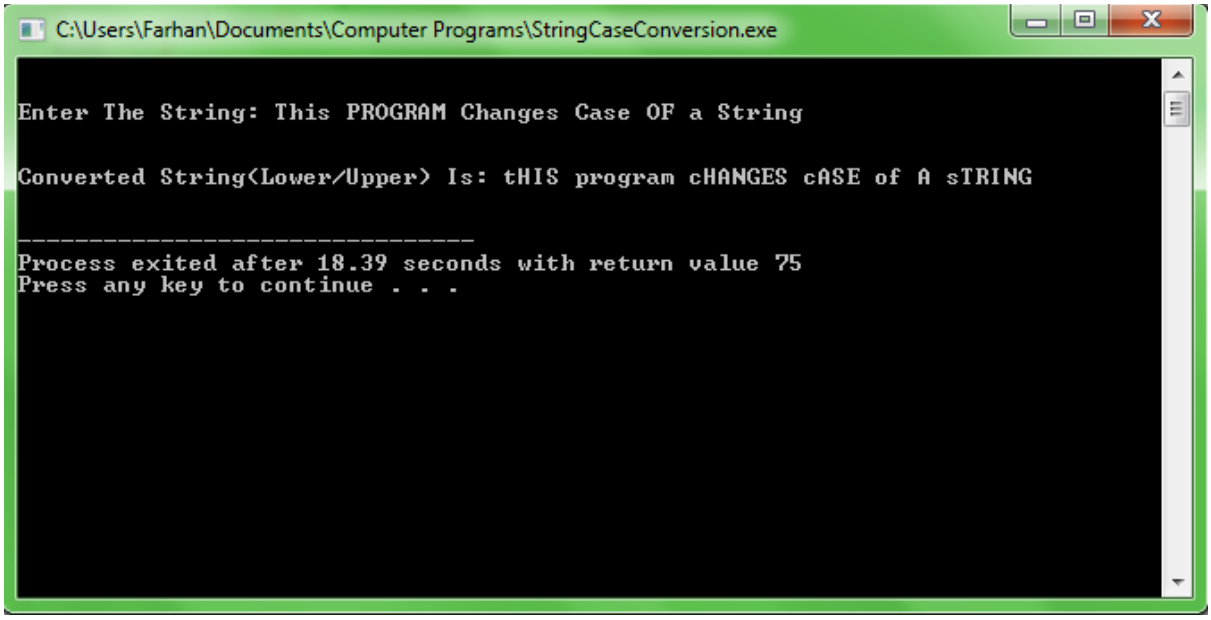
3. Traverse string character one by one till the length of string
   If the letter is upper
   New-string= letter + 32
   If the letter is lower,
   New-string= letter – 32
   If String reached at the end
   Break
4. Print a new string.

CODE:

```
#include<stdio.h>

  #include<string.h>

  void main()

  {



  char str[100];

  int i;



  printf("\n\nEnter The String: ");

  gets(str);



 for (i=0;i<=strlen(str);i++)

  {

  if (str[i]>='A'&&str[i]<='Z')

  {

  str[i] = str[i] + 32;
```

```
  }
  else if (str[i] >= 'a' && str[i] <= 'z')
  {
  str[i] = str[i] - 32;
  }
  }

printf("\n\nConverted String(Lower/Upper) Is: %s\n\n",str);

  }
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learn how to change the string from lower to upper case and vice versa by using ASCII value and changing them accordingly.

PROGRAM-25

OBJECT:

Program for addition of two 3x3 matrices.

INTRODUCTION:

The program prompts the user to enter elements of the first matrix and then the elements of the other matrix. It then adds the corresponding elements and displays the addition matrix as the output.

The addition matrix will also have 3 rows and 3 columns.

ALGORITHM:

1. START
2. Create 3 2-d integer arrays viz., first, second and sum
   First two arrays to store the values entered by the user and sum array to store the final addition matrix.
3. Declare variables c and d of type integer
4. Input the values
5. Use two for loops add the corresponding elements of the two arrays and store them at their respective positions in the sum array.
6. Print the elements of the sum array
7. END

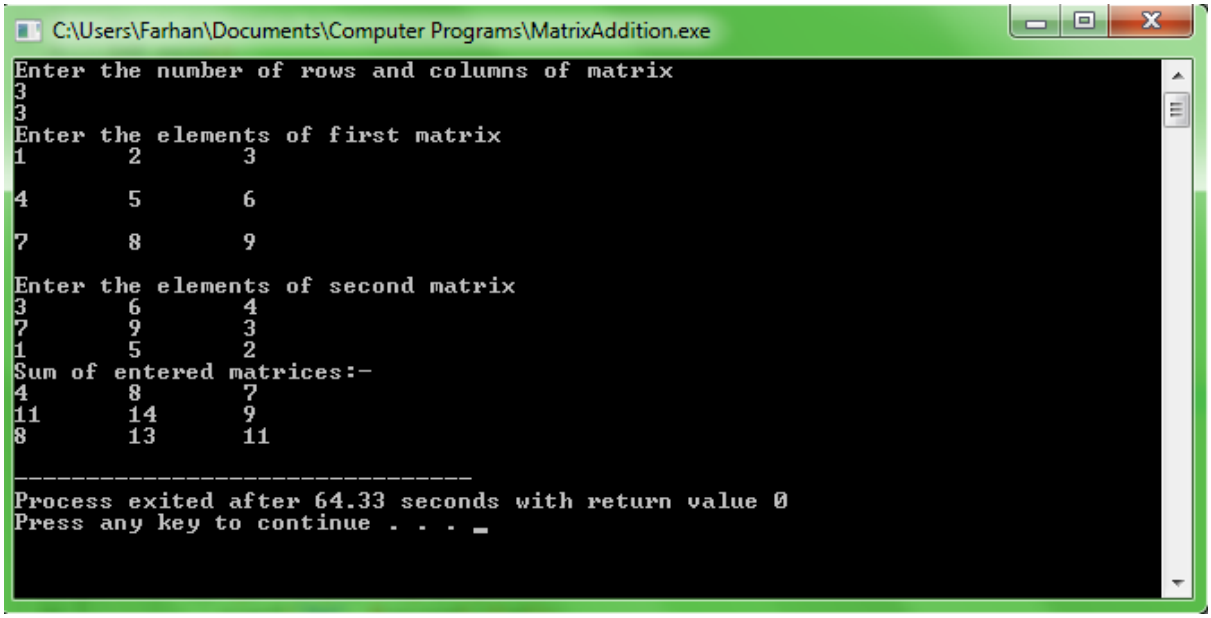
CODE:

```
#include <stdio.h>


int main()

{

  int m, n, c, d, first[10][10], second[10][10], sum[10][10];
```

```
printf("Enter the number of rows and columns of matrix\n");
scanf("%d%d", &m, &n);
printf("Enter the elements of first matrix\n");

for (c = 0; c < m; c++)
{
        for (d = 0; d < n; d++)
    {
            scanf("%d", &first[c][d]);

    }
   printf("\n");
 }
printf("Enter the elements of second matrix\n");

for (c = 0; c < m; c++)
  for (d = 0 ; d < n; d++)
    scanf("%d", &second[c][d]);

printf("Sum of entered matrices:-\n");

for (c = 0; c < m; c++) {
```

```
    for (d = 0 ; d < n; d++) {
      sum[c][d] = first[c][d] + second[c][d];
      printf("%d\t", sum[c][d]);
    }
    printf("\n");
  }
  return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learned how to create a 2-d array. We also learn how to store values in a 2-d array and also how to add the corresponding elements of the two arrays and store the array in another array.

PROGRAM-26

OBJECT:

Program to multiply two 3x3 matrices.

INTRODUCTION:

This program asks the user to enter the size (rows and columns) of two matrices.

To multiply two matrices, **the number of columns of the first matrix should be equal to the number of rows of the second matrix**.

The program below asks for the number of rows and columns of two matrices until the above condition is satisfied.

Then, the multiplication of two matrices is performed, and the result is displayed on the screen.

ALGORITHM:

Step 1: START

Step 2: Enter the row and column of the first (a) matrix.

Step 3: Enter the row and column of the second (b) matrix.

Step 4: Enter the elements of the first (a) matrix.

Step 5: Enter the elements of the second (b) matrix.

Step 6: Set a loop up to row.

Step 7: Set an inner loop up to the column.

Step 8: Set another inner loop up to the column.

Step 9: Multiply the first (a) and second (b) matrix and store the element in the third matrix (c)

Step 10: Print the final matrix.

Step 11: END

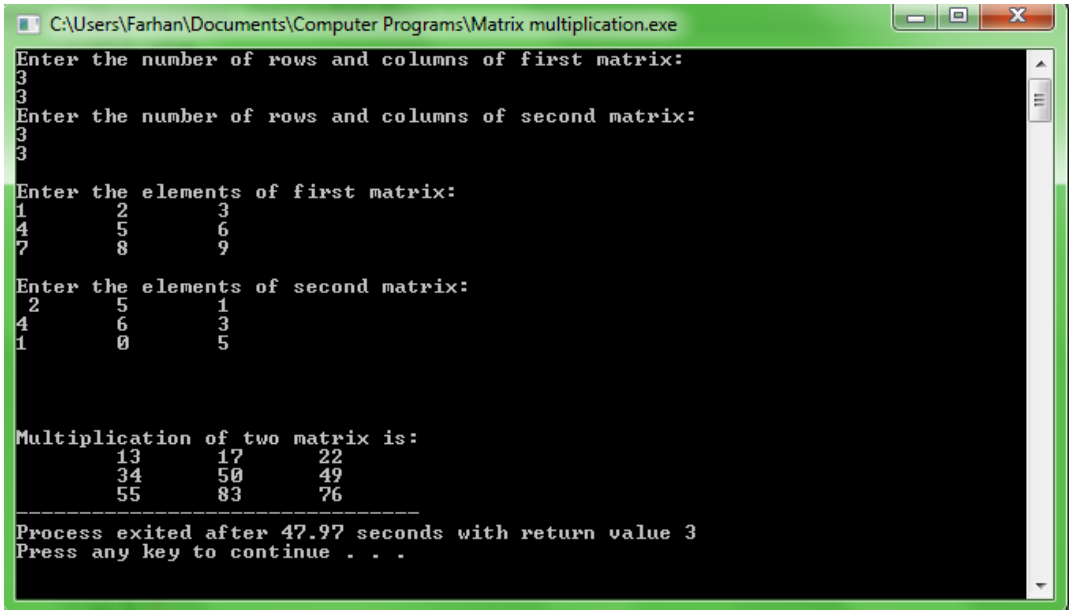
CODE:

```
#include <stdio.h>

void main()

{

int a[25][25],b[25][25],c[25][25],i,j,k,r,s;

int m,n;

printf("Enter the number of rows and columns of first matrix:\n");

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

printf("Enter the number of rows and columns of second matrix:\n");

scanf("%d%d",&r,&s);

if(m!=r)

printf("\n The matrix cannot multiplied");

else

{

printf("\nEnter the elements of first matrix:\n");

for(i= 0;i<m;i++)

{

for(j=0;j<n;j++)

scanf("\t%d",&a[i][j]);

}

printf("\nEnter the elements of second matrix:\n ");
```

```
for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

scanf("\t%d",&b[i][j]);

}

for(i=0;i<m;i++)

{

printf("\n");

for(j=0;j<n;j++)

{

c[i][j]=0;

for(k=0;k<m;k++)

c[i][j]=c[i][j]+a[i][k]*b[k][j];

}

}

}

printf("\nMultiplication of two matrix is:");

for(i=0;i<m;i++)

{

printf("\n");

for(j=0;j<n;j++)

printf("\t%d",c[i][j]);

}
```

```
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learned how to create a 2-d array. We also learn how to store values in a 2-d array and also how to multiply elements of the two matrix arrays and store the array in another array.

PROGRAM-27

OBJECT:

Program to swap two numbers using pointers.

INTRODUCTION:

The program prompts the user to enter 2 numbers and the swaps numbers using pointers
Pointer: Pointer in C language is a variable that stores/points the address of another variable. A pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type.
Pointer syntax: data_type *var_name;
where * is used to denote that var_name is a pointer variable.

ALGORITHM:

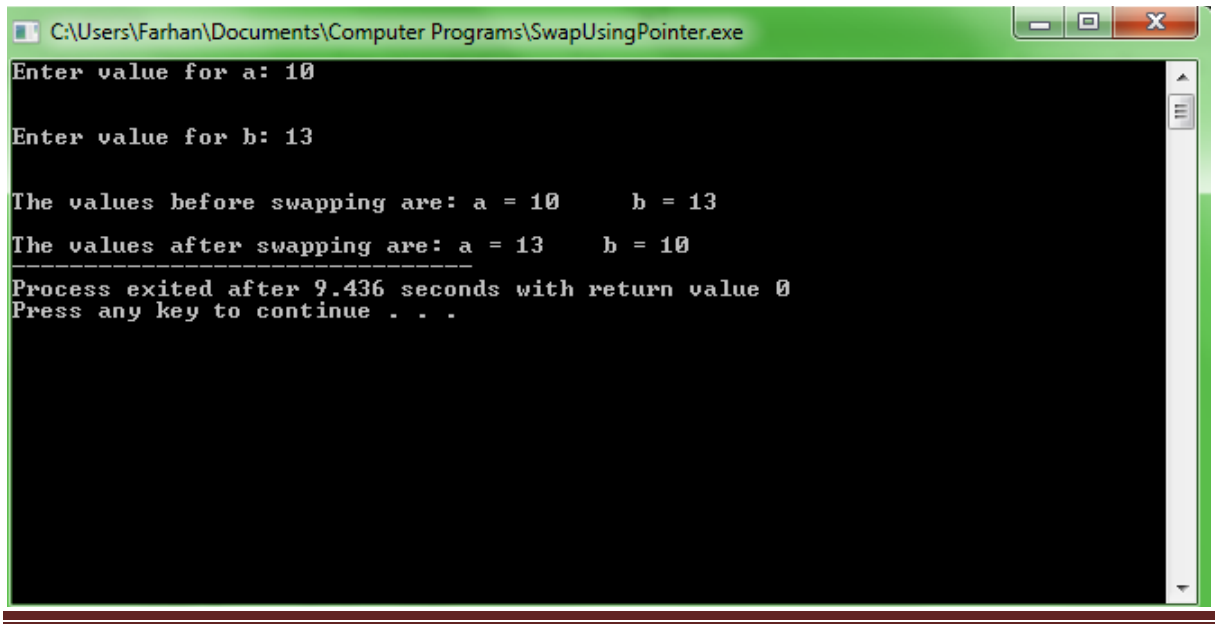
1. START
2. Declare variables a and b of integer type
3. Input the numbers
4. Store the location of a and b inpointer variables ptra and ptrb respectively
5. Perform following algorithm:
   temp = *ptra; temp stores the value at location ptra
   *ptra = *ptrb; assigning value at location ptrb to ptra
   *ptrb = temp; assign value of themp to the variable at location ptrb
6. Print the values of a and b
7. END.

CODE:

```
#include<stdio.h>

int main()
{

   int a, b;
   int *ptra, *ptrb;
```

```
    int temp;

    printf("Enter value for a: ");
    scanf("%d", &a);

    printf("\n\nEnter value for b: ");
    scanf("%d", &b);

    printf("\n\nThe values before swapping are: a = %d     b = %d", a, b);

    ptra = &a;
    ptrb = &b;

    temp = *ptra;
    *ptra = *ptrb;
    *ptrb = temp;

 printf("\n\nThe values after swapping are: a = %d    b = %d", a, b);

       return 0;

}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to declare and use pointers. We also learned how to swap numbers using pointers and how to pass address of variables to pointers.

PROGRAM-28

OBJECT:

Programs to generate the employee details using structures.

INTRODUCTION:

Structure is commonly reffered to as user-defined data type. Structure is similar to an array but the only difference is that array is collection of similar data type on the other hand structure is collection of different data type. A structure can contain any data type including array and another structure as well. Each variable declared inside structure is called member of structure.

o store employees details we used structure array. Each element in the structure array represents the single employee.

Each Structure i.e. *Employee* contains:

- Name
- Id
- Salary
- Age

After taking input, we simply printed all employee details by iterating using for loop.

ALGORITHM:

1. START
2. Define structure employee with required data in it
3. Enter number of employees
4. Enter details of name, age, id, salary.
5. Then print the details using dot('.') notation
6. END

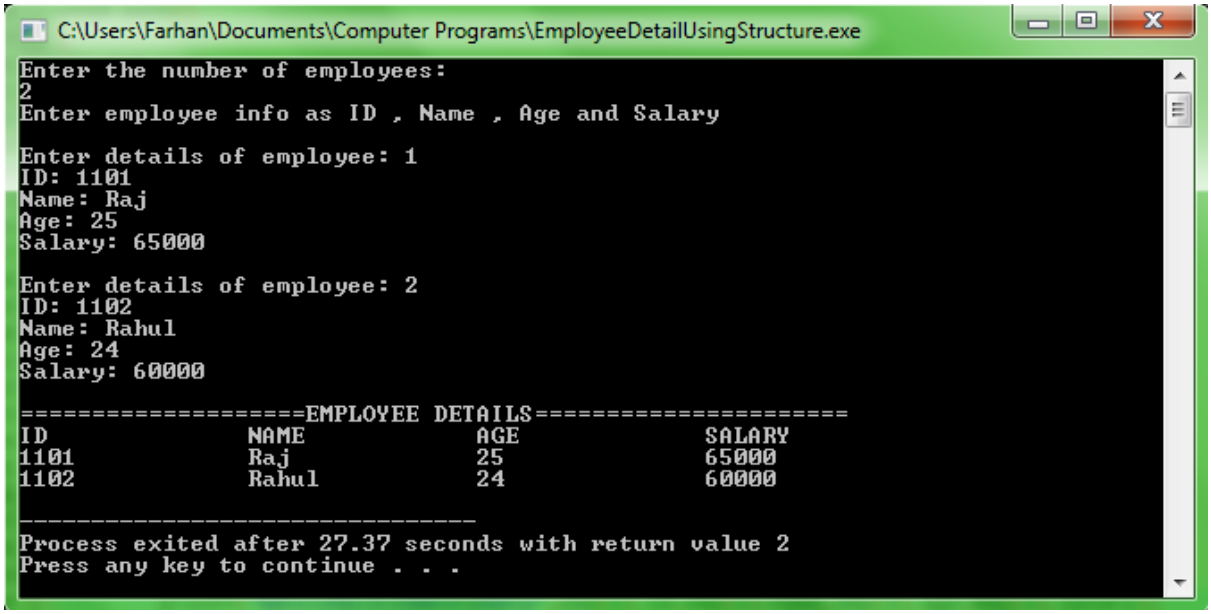
CODE:

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

```
struct employee
{
   int id,age,salary;
   char name[25];
}emp[100];


void main()
{
   int i,n;
   printf("Enter the number of employees:\n");
   scanf("%d",&n);
   printf("Enter employee info as ID , Name , Age and Salary\n");
   for(i=0;i<n;i++)
   {
      printf("\nEnter details of employee: %d\n", i+1);
         printf("ID: ");
               scanf("%d",&emp[i].id);
               printf("Name: ");
               scanf("%s",&emp[i].name);
               printf("Age: ");
               scanf("%d",&emp[i].age);
               printf("Salary: ");
               scanf("%d",&emp[i].salary);
```

```
    }
    printf("\n--------------------EMPLOYEE DETAILS----------------------");
      printf("\nID\t\tNAME\t\tAGE\t\tSALARY\n");
    for(i=0;i<n;i++)
    {
       printf("%d\t\t%s\t\t%d\t\t%d\n",emp[i].id,emp[i].name,emp[i].age,emp[i].salary);
    }
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learned about structures in C and about its use as well.

We learned how to create and store values in it.

PROGRAM-29

OBJECT:

Program to find area and perimeter of a circle, rectangle, square and triangle using functions.

INTRODUCTION:

This program is used to find the perimeter of a circle, rectangle and triangle. The formula used in this program are:
perimeter of rectangle: 2 * (a + b)

Area of rectangle: a*b
perimeter of triangle: a + b + c
area of triangle: $(s(s-a)(s-b)(s-c))^{1/2}$

perimeter of circle: 2 * pi * r

area of circle: 3.14*r*r

ALGORITHM:

1. START
2. Create functions for area and perimeter of different figures of type float with required variables
3. Declare variables length, breadth, radius, area, perimeter, a, b, c of type float
4. Take input regarding the figure
5. Using switch statement, calculate the area and perimeter of the required figure by passing the values to the functions.
6. Display the values of area and perimeter
7. END

CODE:

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

```
#include<math.h>
#include<stdlib.h>
float per_circle(float r);
float area_circle(float r);
float per_rectangle(float l, float b);
float area_rectangle(float l, float b);
float per_square(float l);
float area_square(float l);
float per_triangle(float a, float b, float c);
float area_triangle(float a, float b, float c);
void main()
{
        while(1)
        {
        float length, breadth,side, a, b, c, radius;
        float area, perimeter;
        int fig_code;
        printf("\n=======================================\n");
        printf("1 => Circle\n");
        printf("2 => Rectangle\n");
        printf("3 => Square\n");
        printf("4 => Triangle\n");
        printf("\nPress 0 to exit!");
```

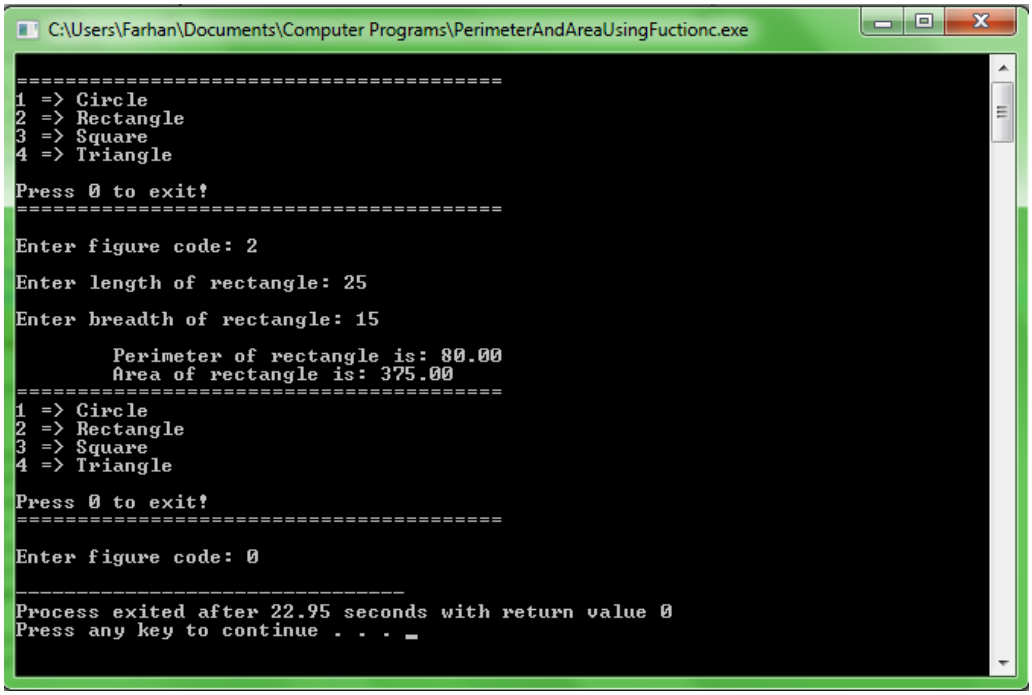
```
printf("\n=======================================\n");
printf("\nEnter figure code: ");
scanf("%d", &fig_code);

switch(fig_code)
{
    case 1:
        printf("\nEnter the radius of the circle: ");
        scanf("%f",&radius);
        perimeter = per_circle(radius);
        area = area_circle(radius);
        printf("\n\tPerimeter of circle is: %.2f", perimeter);
        printf("\n\tArea of circle is: %.2f", area);
        break;
    case 2:
        printf("\nEnter length of rectangle: ");
        scanf("%f", &length);
        printf("\nEnter breadth of rectangle: ");
        scanf("%f", &breadth);
        perimeter = per_rectangle(length, breadth);
        area = area_rectangle(length, breadth);
        printf("\n\tPerimeter of rectangle is: %.2f", perimeter);
        printf("\n\tArea of rectangle is: %.2f", area);
```

```
                break;
        case 3:
                printf("\nEnter side of square: ");
                scanf("%f", &side);
                perimeter = per_square(side);
                area = area_square(side);
                printf("\n\tPerimeter of square is: %.2f", perimeter);
                printf("\n\tArea of square is: %.2f", area);
                break;
        case 4:
                printf("\nEnter the sides of triangle:\n");
                scanf("%f %f %f", &a, &b, &c);
                perimeter = per_triangle(a, b, c);
                area = area_triangle(a, b, c);
                printf("\n\tPerimeter of triangle is: %f.2", perimeter);
                printf("\n\tArea of triangle is: %.2f", area);
                break;
        case 0:
                exit(0);
                break;
        default:
                printf("\nPlease enter a valid code!!");
    }
```

```
  }
}
float per_circle(float r)
{
        float perimeter = 2*3.14*r;
        return perimeter;
}
float area_circle(float r)
{
        float area = 3.14*r*r;
}
float per_rectangle(float l, float b)
{
        float perimeter = 2*(l+b);
        return perimeter;
}
float area_rectangle(float l, float b)
{
        float area = l*b;
        return area;
}
float per_square(float l)
{
```

```
	float perimeter = 4*l;
	return perimeter;
}
float area_square(float l)
{
	float area = l*l;
	return area;
}
float per_triangle(float a, float b, float c)
{
	float perimeter = a+b+c;
	return perimeter;
}
float area_triangle(float a, float b, float c)
{
	float s = 0.5*(a+b+c);
	float area = sqrt(s*(s-a)*(s-b)*(s-c));
	return area;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned more about switch case statements and functions together in a program. We also learned about passing variables to a function and returning the value from the function and storing it in another variable.

PROGRAM-30

OBJECT:

Write a program to pass and return pointer to a function and hence calculate the average of an array.

INTRODUCTION:

The program prompts the user to enter values of elements of the array and calculates the average of the array by using pointers and displays it as the output.

ALGORITHM:

1. START
2. Create a pointer variable a, which points to an integer data.
3. Now, create another variable n (size of array) and a variable total=0, which will stpre total sum of all the elements of the array.
4. Now create size_of_array(n)*size_of_each_element(integer) times space using malloc() function, assigning the starting address of this space to the pointer variable, a.
5. Using for loop (0 to n), start iteration, reach at every array element location by adding the value of I to the value of a, taking array element as input.
6. Again start iteration, this time reach at each array element location, and accessing them to add to the total variable
7. Using total, calculate the average of the array
8. Print average
9. END

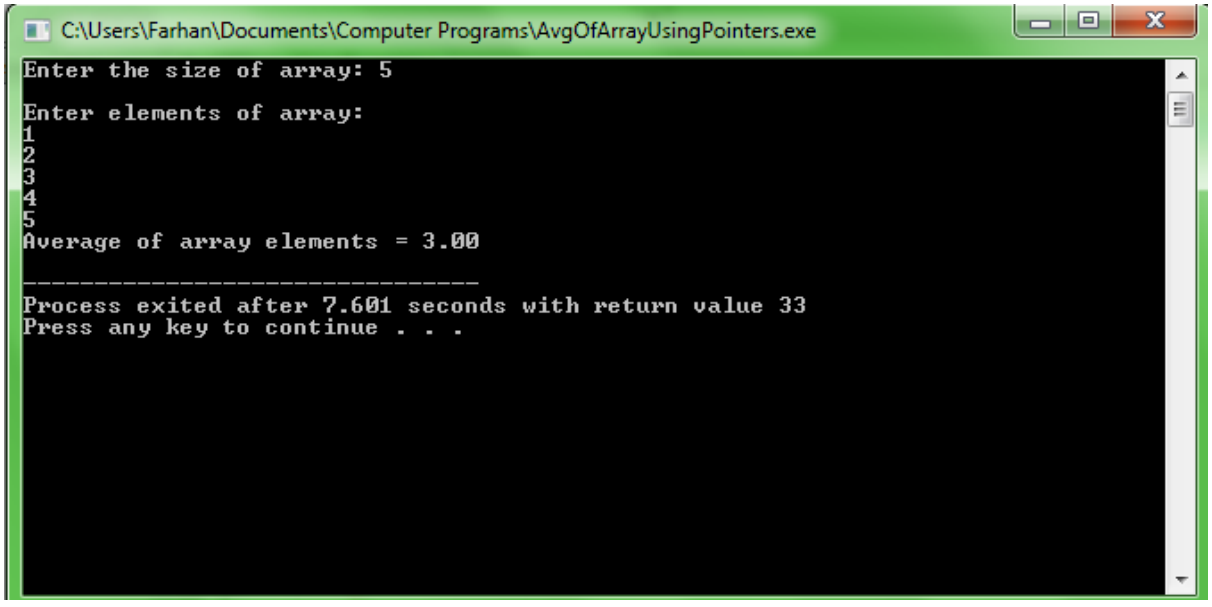
CODE:

```
/*
 C program to find the sum of all elements of an array using pointers as arguments.
*/
```

```
#include <stdio.h>

#include<malloc.h>

void main()

{

    int array[20];

    int i,total=0,n;

    int *a;

    printf("Enter the size of array: ");

    scanf("%d",&n);

    a = (int *)malloc(n*sizeof(int));

    printf("\nEnter elements of array: \n");

    for(i=0; i < n; i++)

    scanf("%d",a+i);

            for (i = 0; i < n; i++)

        {
```

```
            total += *(a + i);


    }
      float avg = total/n;


     printf("Average of array elements = %.2f\n", avg);


}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learned about pointers. We learn how to calculate the sum of elements of an array using pointers. We also learn about the malloc function. In C, the library function malloc is used to allocate a block of memory on the heap. The program accesses this block of memory via a pointer that malloc returns.

PROGRAM-31

OBJECT:

Write a program to pass an array as pointer to a function that calculates the sum of all the elements of the array.

INTRODUCTION:

Just like variables, array can also be passed to a function as an argument using pointers.

The program calls a function to add all the element of an array and passes the array argument as a pointer. The program should dynamically allocate a piece of memory for that array and use a pointer to point to that array memory as well as traverse that array using a pointer.

ALGORITHM:

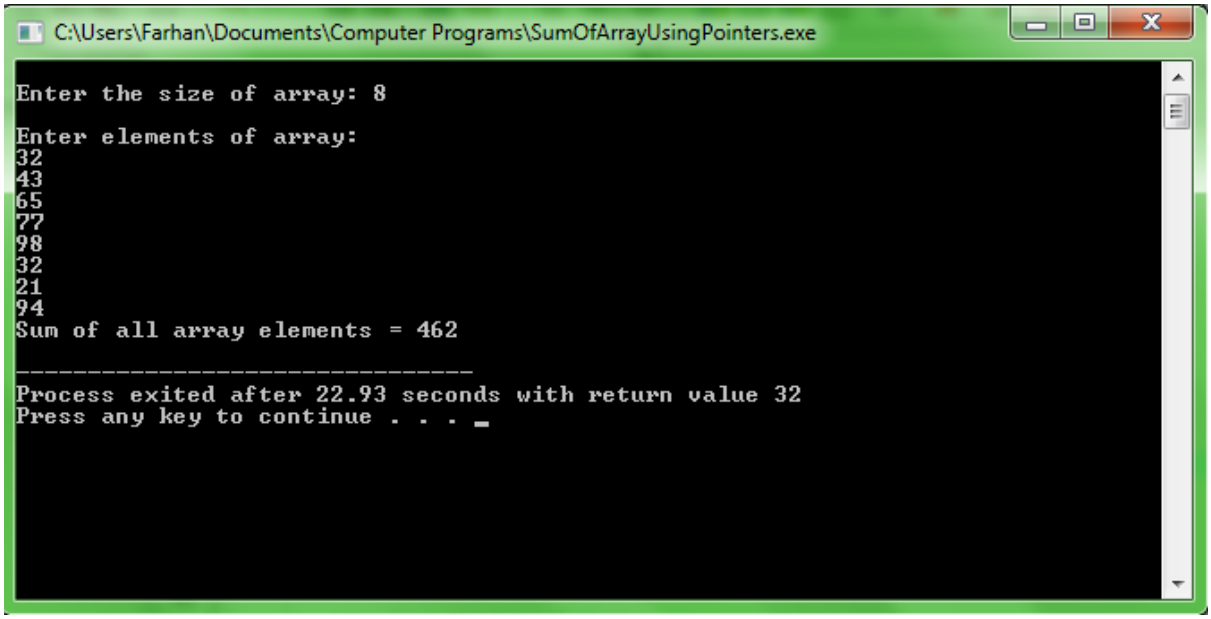
1. START
2. Declare an array of some size.
3. Take a variable sum, which will store the total sum of all the array elements.
4. Create a function with a single argument, in which we will pass the above created array argument. This function will return the sum.
5. Inside this function, a for loop will run from 0 to array size-1, accessing each element of array by adding the iterator i to the pointer variable (array argument, which was passed to this function).
6. Elements will get add and the total sum will be returned from this function.
7. Variable sum inside main() function, will get this returned value and will be printed.
8. END

CODE:

```
#include <stdio.h>

void main()

{
```

```
        int array[20];
        int i, n, sum;
         printf("\nEnter the size of array: ");
        scanf("%d",&n);
        printf("\nEnter elements of array: \n");
        for(i=0; i<n; i++)
        {
         scanf("%d",&array[i]);
   }
        int addnum(int *ptr, int n);
        sum = addnum(array, n);

      printf("Sum of all array elements = %d\n", sum);
}
int addnum(int *ptr, int n)
{
        int index, total = 0;
        for (index = 0; index < n; index++)
         {
        total += *(ptr + index);
        }
        return(total);
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned how to pass array to a function using pointers and how to calculate sum of elements of array using this method.

PROGRAM-32

OBJECT:

Write a program to demonstrate the example of array of pointers.

INTRODUCTION:

As we know that, pointers are the special type of variables that are used to store the address of another variable. And array is the group of similar type of variables (using single name for all variables), that takes contiguous memory locations.

"Array of pointes" is an array of the pointer variables. Here, in this program, we are declaring an array of float pointers that will store the address of float variables.

Assign address to array of pointers

This step is similar to any other pointer variable. We get the address of a variable using the **address of** & operator and then save that address in the array of pointers.

ALGORITHM:

1. START
2. Declare five variables to store marks of five subjects
3. Declare an array of integer pointers that will point to the address of variables containing marks of different subjects
4. Take input of the marks form user
5. Calculate percentage of each subject using pointers
6. Print percentage of marks
7. END

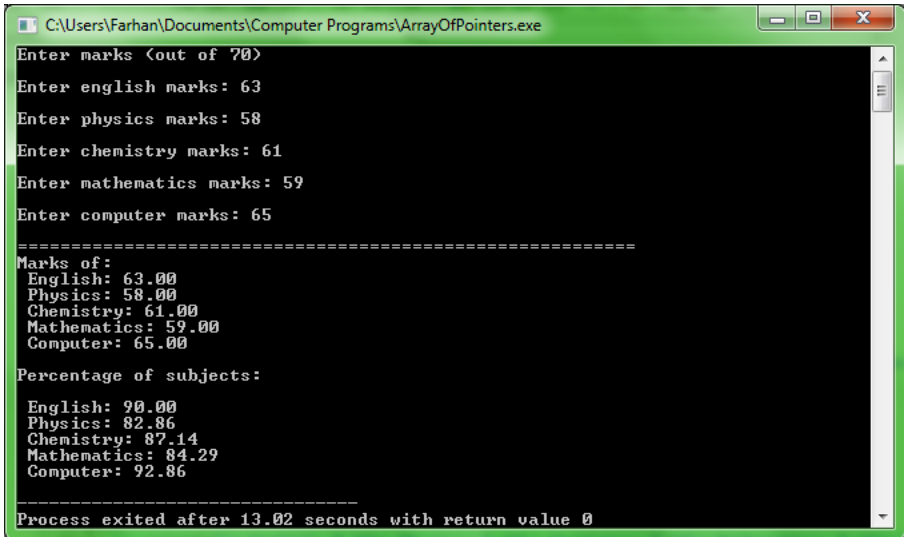
CODE:

```
/*C program to demonstrate example of array of pointers.*/

#include <stdio.h>



int main()
```

```
{
  /*declare same type of variables*/
  float eng,phy,chem,math,comp;
  int i;
  /*we can create an integer pointer array to
 store the address of these integer variables*/
  float *ptr[5];

  /*assign the address of all integer variables to ptr*/
  ptr[0]= &eng;
  ptr[1]= &phy;
  ptr[2]= &chem;
  ptr[3]= &math;
  ptr[4]= &comp;

  /*assign the values to variables*/
 printf("Enter marks (out of 70)\n");
 printf("\nEnter english marks: ");
 scanf("%f",&eng);
  printf("\nEnter physics marks: ");
 scanf("%f",&phy);
  printf("\nEnter chemistry marks: ");
 scanf("%f",&chem);
  printf("\nEnter mathematics marks: ");
```

```
  scanf("%f",&math);
   printf("\nEnter computer marks: ");
  scanf("%f",&comp);
  printf("\n=========================================================\n");

   /*print values using pointer variable*/
   printf("Marks of:\n English: %.2f\n Physics: %.2f\n Chemistry: %.2f\n Mathematics: %.2f\n
Computer: %.2f\n",*ptr[0],*ptr[1],*ptr[2],*ptr[3],*ptr[4]);

   /*calculating percentage of all subjects using pointer*/
   for(i=0; i<5; i++)
   {
          *ptr[i] = (*ptr[i]/70)*100;
          }
          printf("\nPercentage of subjects:\n");

   printf("\n English: %.2f\n Physics: %.2f\n Chemistry: %.2\n Mathematics: %.2f\n Computer:
%.2f\n",*ptr[0],*ptr[1],*ptr[2],*ptr[3],*ptr[4]);
  return 0;
}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

We learned about array of pointers and how to store address of variables in it. We also learned how to perform basic mathematical operations using array of pointers.

PROGRAM-33

OBJECT:

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

INTRODUCTION:

This C Program creates a file & store information. We frequently use files for storing information which can be processed by our programs. In order to store information permanently and retrieve it we need to use files and this program demonstrate file creation and writing data in that.

ALGORITHM:

1. START
2. Declare a file pointer
3. Open file named emp.txt using fopen() in writing mode
4. If file pointer points to a null value, print file doesn't exist
5. Enter details of name, age, and salary
6. Close the file using fclose()
7. END

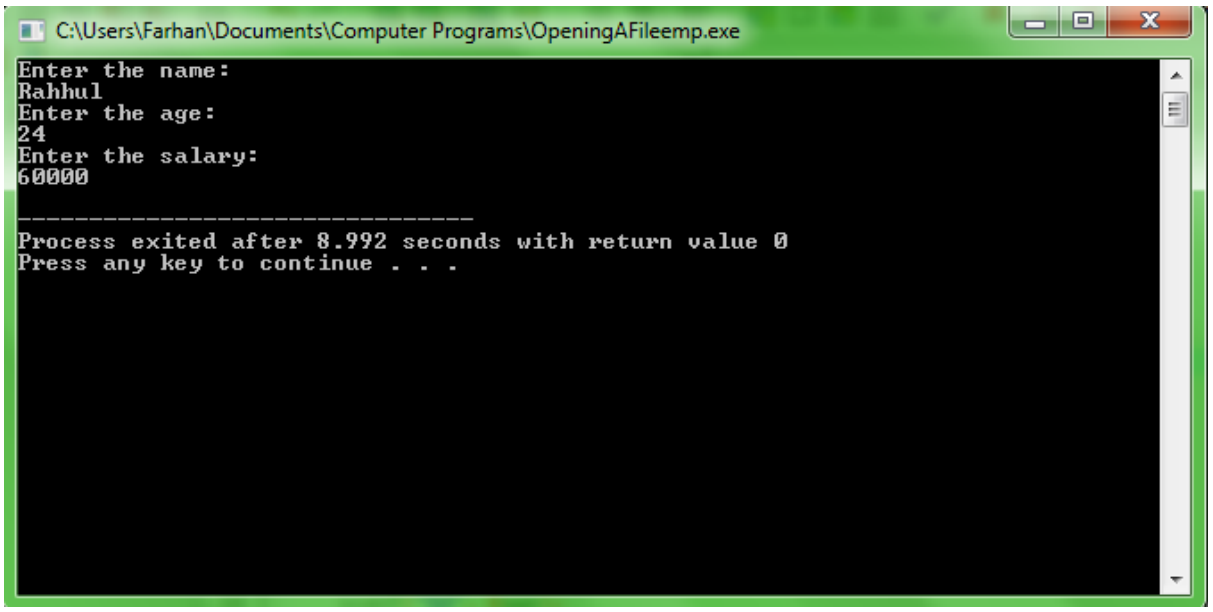
CODE:

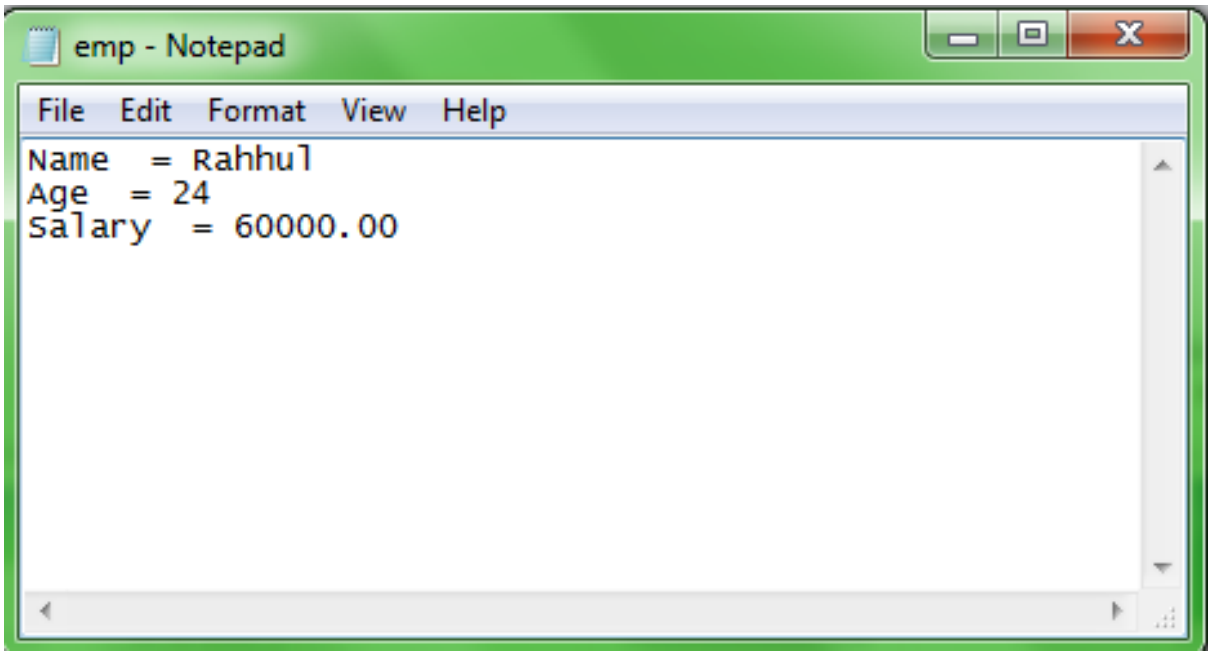
```
#include<stdio.h>

#include<conio.h>



void main()

{

   FILE *fptr;

   char name[20];
```

```
int age;

float salary;



/* open for writing */

fptr = fopen("emp.txt", "w");



if (fptr == NULL)

{

   printf("File does not exist!!\n");

   return;

}

printf("Enter the name:\n");

scanf("%s", name);

fprintf(fptr, "Name  = %s\n", name);



printf("Enter the age:\n");

scanf("%d", &age);

fprintf(fptr, "Age  = %d\n", age);



printf("Enter the salary:\n");

scanf("%f", &salary);

fprintf(fptr, "Salary  = %.2f\n", salary);
```

```
    fclose(fptr);

}
```

OUTPUT:





LEARNING FROM EXPERIMENT:

From this experiment we learned about basic file handling in C language.

We learned how to open and close a file and how to create a text file. We learned about functions like fopen, fclose and file pointers. Moreover, we also learned how to write data in a file using various funtions.

PROGRAM-34

OBJECT:

Write a program which copies one file contents to another.

INTRODUCTION:

A File can be used to store a large volume of persistent data. Like many other languages 'C' provides the following file management functions,

- Creation of a file
- Opening a file
- Reading a file
- Writing to a file
- Closing a file

In this program we will copy a file to another file, firstly you will specify a file to copy. We will open the file and then read the file that we wish to copy in "read" mode and target file in "write" mode.

ALGORITHM:

1. START
2. Input file path of source and destination file.
3. Open source file in r (read) and destination file in w (write) mode.
4. Read character from source file and write it to destination file using fputc().
5. Repeat step 3 till source file has reached end.
6. Close both source and destination file.
7. END

CODE:

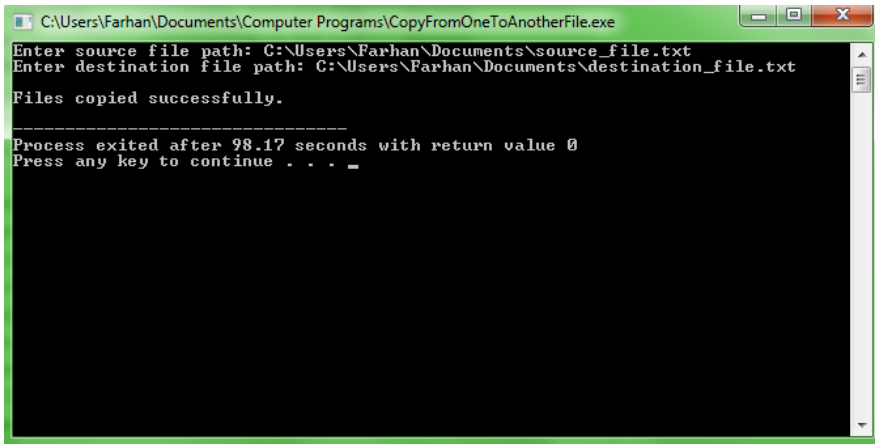
```
//C program to copy contents of one file to another.

#include <stdio.h>

#include <stdlib.h>
```

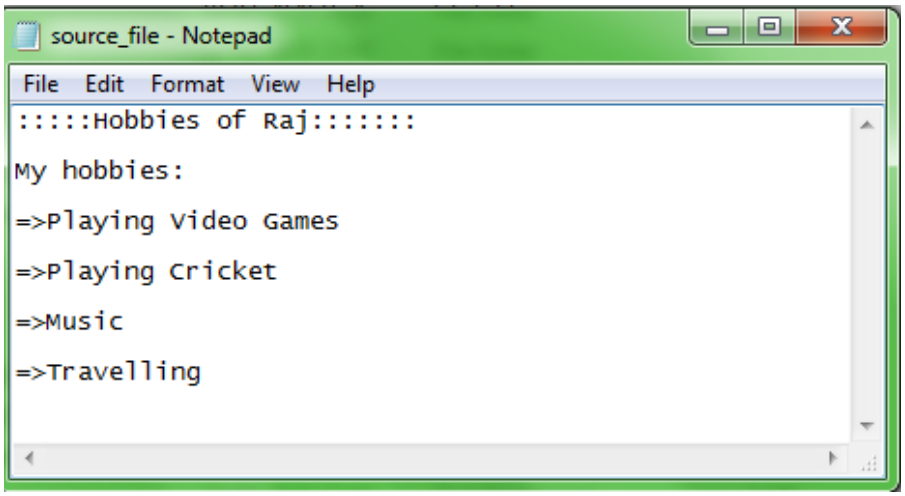
```
int main()
{
    FILE *sourceFile;
    FILE *destFile;
    char sourcePath[100];
    char destPath[100];

    char ch;

    /* Input path of files to copy */
    printf("Enter source file path: ");
    scanf("%s", sourcePath);
    printf("Enter destination file path: ");
    scanf("%s", destPath);

    /*
     * Open source file in 'r' and
     * destination file in 'w' mode
     */
    sourceFile  = fopen(sourcePath, "r");
    destFile    = fopen(destPath,   "w");
```

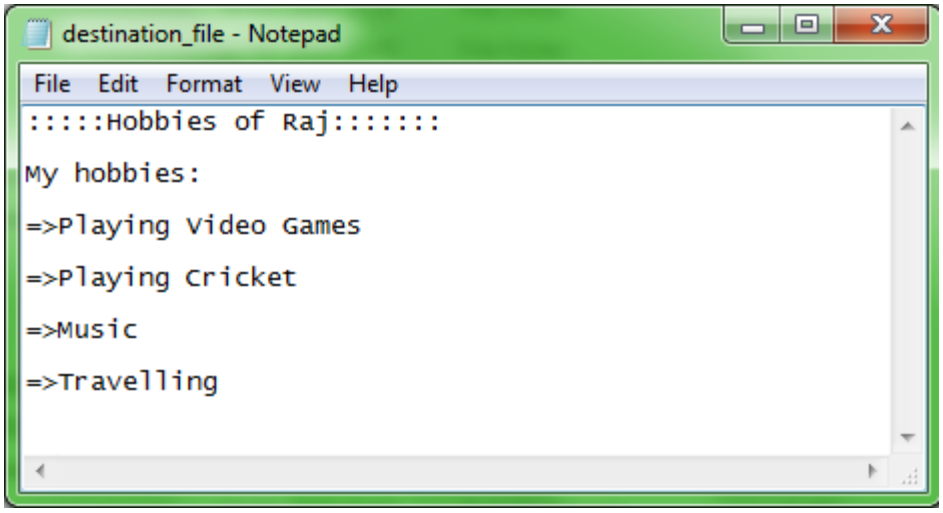
```
/* fopen() return NULL if unable to open file in given mode. */
if (sourceFile == NULL || destFile == NULL)
{
    /* Unable to open file hence exit */
    printf("\nUnable to open file.\n");
    printf("Please check if file exists and you entered correct path.\n");

    exit(EXIT_FAILURE);
}
 // Copy file contents character by character.
ch = fgetc(sourceFile);
while (ch != EOF)
{
    // Write to destination file
    fputc(ch, destFile);

    // Read next character from source file
    ch = fgetc(sourceFile);
}
printf("\nFiles copied successfully.\n");

// Finally close files to release resources
```

```
    fclose(sourceFile);

    fclose(destFile);

    return 0;

}
```

OUTPUT:







LEARNING FROM EXPERIMENT:

We learned about file handling of .txt files in C program. We also learned how to copy content from one file to another.

PROGRAM-35

OBJECT:

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

INTRODUCTION:

Given a text file, our task is to convert all the lower case characters of file into upper case.

Open the file in **read** mode. Check if there is any error in opening or locating a file. If yes, then throw an error message.

If the file is found, then with the help of while loop, convert all the characters using toupper of that file into upper case.
Close the file using **fclose()** function by passing the file pointer in it.

ALGORITHM:

1. START
2. Open source file in read mode, store its reference in fptr.
3. Create a temporary file to store result, store its reference in dest.
4. Read a character from source file fptr. Store character read in ch
5. Convert lowercase characters to uppercase. Which means if islower(ch) then convert to uppercase.
6. Write converted character to dest file.
7. Repeat step 3-5 till end of file.
8. Close both files fptr as well as dest.
9. END

CODE:

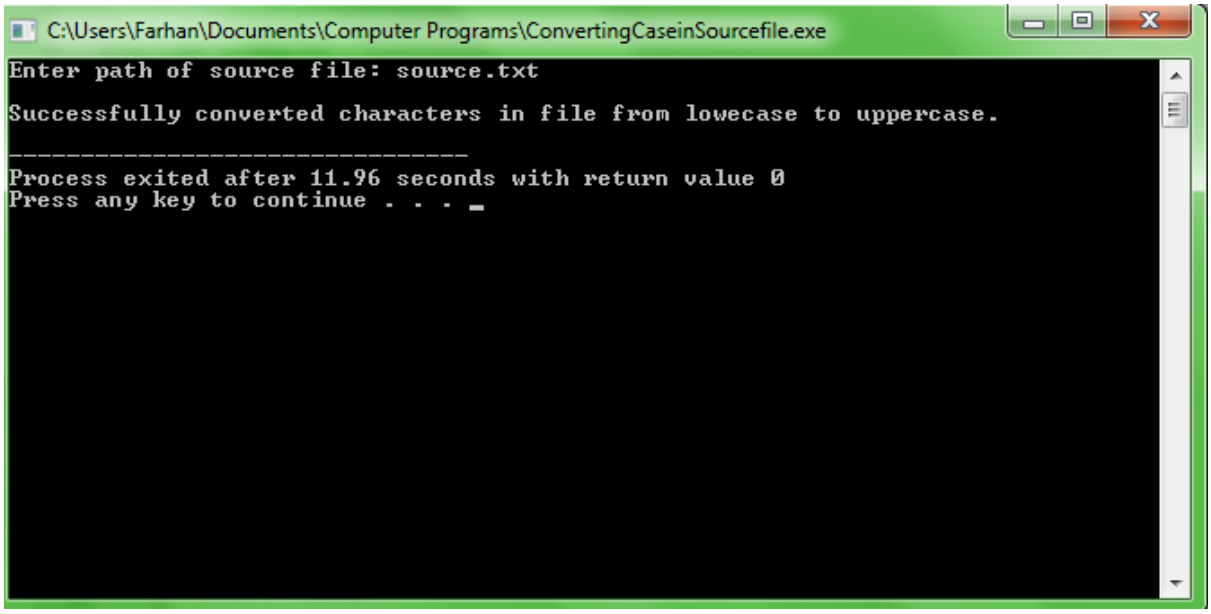
```
/*C program to convert lowercase to uppercase of a file and copy the contents after converting case

 copying it to another file*/
```

```
#include <stdio.h>

#include <stdlib.h>



void toggleCase(FILE *fptr, const char *path);

int main()

{

   FILE *fPtr;

   char path[100];



   printf("Enter path of source file: ");

   scanf("%s", path);

   fPtr = fopen(path, "r");

   if (fPtr == NULL)

   {

      /* Unable to open file hence exit */

      printf("\nUnable to open file.\n");

      printf("Please check whether file exists!\n");

      exit(EXIT_FAILURE);

   }

   toggleCase(fPtr, path);

   printf("\nSuccessfully converted characters in file from lowecase to uppercase.\n");

   return 0;
```

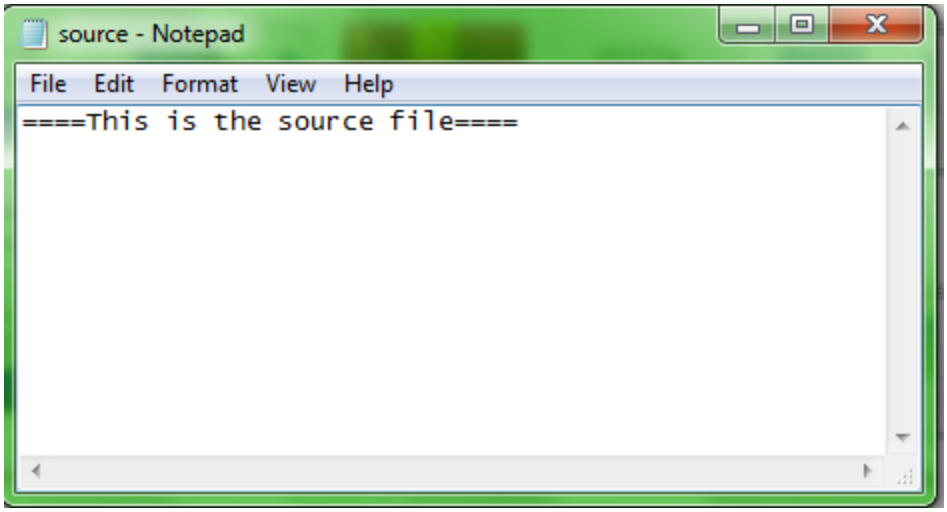
```
}

/**
 * Function to convert lowercase characters to uppercase
 * and uppercase to lowercase in a file.
 */
void toggleCase(FILE *fptr, const char *path)
{
    FILE *dest;
    char ch;


    // Temporary file to store result
    dest = fopen("destination_file.txt", "w");




    // If unable to create temporary file
    if (dest == NULL)
    {
        printf("Unable to convert case.");
        fclose(fptr);
        exit(EXIT_FAILURE);
    }
```

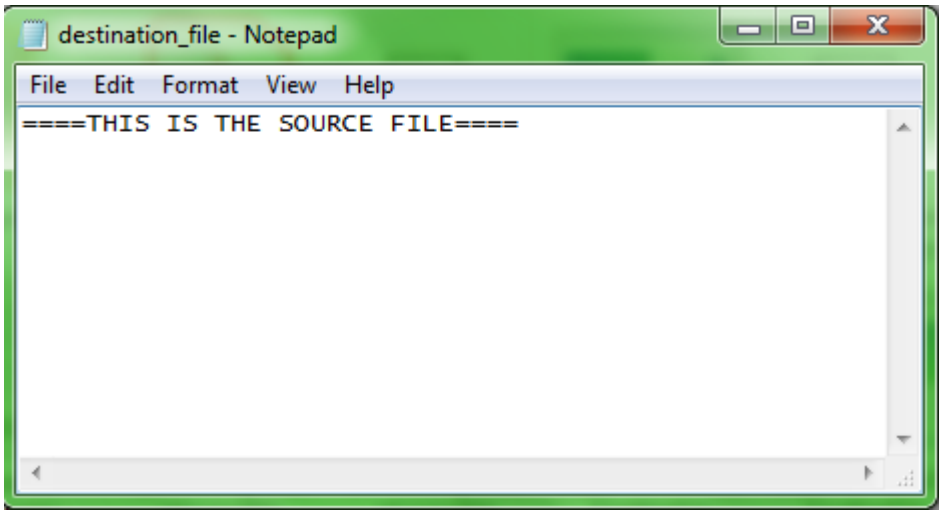
```
    /* Repeat till end of file. */
    while ( (ch = fgetc(fptr)) != EOF)
    {
        if (islower(ch))
            ch = toupper(ch);

        // Print converted character to destination file.
        fputc(ch, dest);
    }

    fclose(fptr);
    fclose(dest);
}
```

OUTPUT:


C:\Users\Farhan\Documents\Computer Programs\ConvertingCaseinSourcefile.exe
Enter path of source file: source.txt
Successfully converted characters in file from lowecase to uppercase.

Process exited after 11.96 seconds with return value 0
Press any key to continue . . .


source - Notepad
File Edit Format View Help
====This is the source file====


destination_file - Notepad
File Edit Format View Help
====THIS IS THE SOURCE FILE====

LEARNING FROM EXPERIMENT:

From this experiment we learned how to change case of letters in a file using C program. We learned about basic file handling functions in C. We also learned how to open, close, read/write and create a file using C language.

PROGRAM-36

OBJECT:

Write a program to find the size of a given file.

INTRODUCTION:

The function fseek() is used to set the file pointer at the specified position. In general, it is used to move the file pointer to the desired position within the file.

The idea is to use fseek() in C and ftell() in C. Using fseek(), we move file pointer to end, then using ftell(), we find its position which is actually size in bytes.

ALGORITHM:

1. Begin
2. Declare function file_size()
3. Open a file pointer fp in read only mode.
4. If fp is equals to null then
    a. Print "File not found" and return -1.

   Else count the file size.
    b. Close the file.
5. Put the file pointer at the beginning of the file
6. Declare a integer variable result and initialize it with the output of the ftell() function.
7. Close file pointer fp.
8. Return result.
9. End

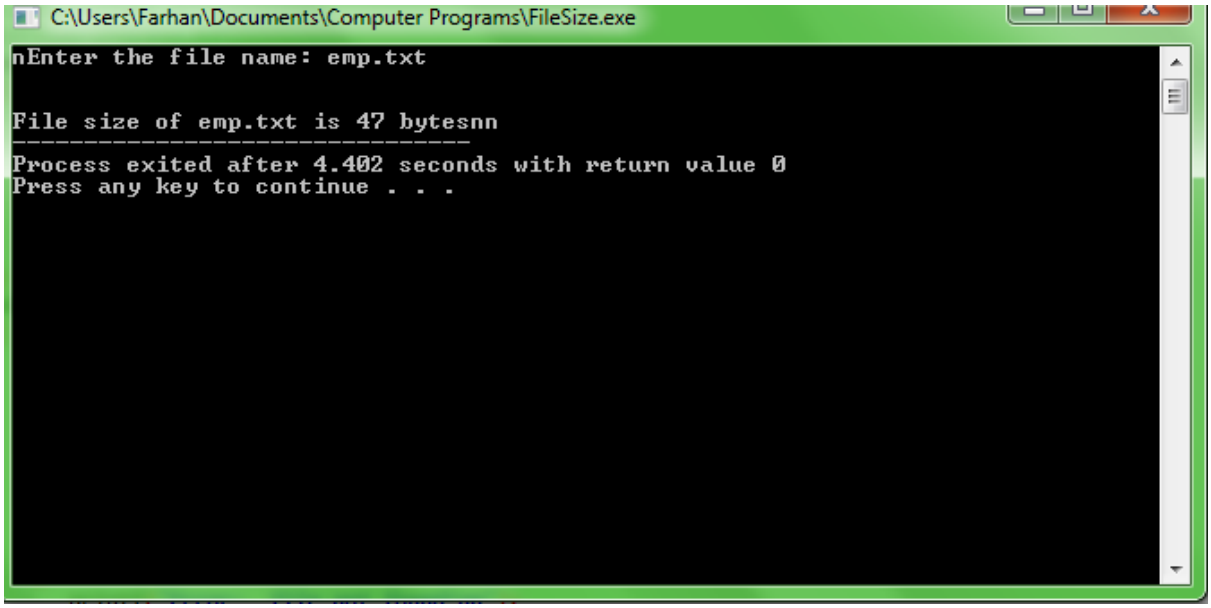
CODE:

```
#include<stdio.h>

#include<process.h>

long file_size(FILE *fp)

 {
```

```
    long int len;
   //move file pointer to end of the file
   fseek(fp, 0L, 2);
   //Obtain the current position of file pointer
   len = ftell(fp);
   return len;
 }
void main(void)
 {
  FILE *fp;
  char filename[20];
  printf("nEnter the file name: ");
  scanf("%s",filename);

  fp = fopen(filename, "r");

  if(fp == NULL)
   {
     printf("Error:  file not found!nn");
     exit(0);
   }
 printf("\n\nFile size of %s is %ld bytesnn",filename, file_size(fp));
```

```
 fclose(fp);

}
```

OUTPUT:



LEARNING FROM EXPERIMENT:

From this experiment we learned about file handling operations and functions. We learned how to find the size of a .txt extension file size in C language.

Program-1

⇒ Objective:

Program to find sum and average of two numbers.

⇒ Introduction:

A natural number is a number that occurs commonly and obviously in nature. The set of natural number, denoted by **N**, can be defined in the following way:-

$\mathbf{N} = \{1, 2, 3, 4, \ldots\}$

In colloquial language, an average is the sum of a list of numbers divided by the number of numbers in the list.

The most common type of average is arithmetic mean. The arithmetic mean or simply called mean of two numbers, such as 3, 5 is obtained by finding a value A such that $3+5 = A+A$ i.e. $A = \frac{3+5}{2} = 4$

If we increase the number of terms to 3, 5 and 7, the arithmetic mean is found by solving the value of A in the eq^n: $3+5+7 = A+A+A$ i.e. $A = \frac{3+5+7}{3} = 5$.

⇒ Algorithm:

Step-1: Start

Step-2: Declare three integer variables a, b, sum and one float variable avg

Step-3: Input two numbers, a, b

Step-4: sum = a+b

Step-5: Print sum

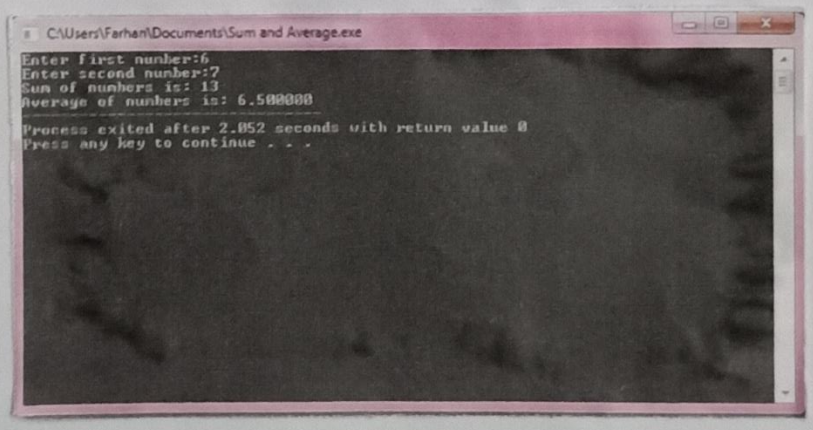
Step-6 avg = $\frac{a+b}{2}$

Step-7 Print avg

Step-8 End.

⇒ Code:

```
#include <stdio.h>
#include<conio.h>
int main()
{
    int a, b, sum;
    float avg;
    printf("Enter first number: ");
    scanf("%d",&a);
    printf("Enter second number");
    scanf("%d", &b);

    sum = a+b;
    printf("Sum of two numbers is=%d", sum);
    avg = (a+b)/2;
    printf("Average of two numbers is = %f", avg);
     return 0;
     getch();
}
```



⇒ Learning from experiment;

We have learned how to input a number and how to perform operations on it.

Program-2

⇒ Objective:

Write a program to find greatest of 10 numbers.

⇒ Introduction:

The program takes the input one by one and at each turn compares the new input with the old input to see which is greater. If the new input is found greater, the old input is discarded and replaced with the new input and program proceed in a similar fashion to accept another input. If old input is found greater, it is retained and the new input is discarded away and the program proceeds to accept the next input.

⇒ Algorithm:

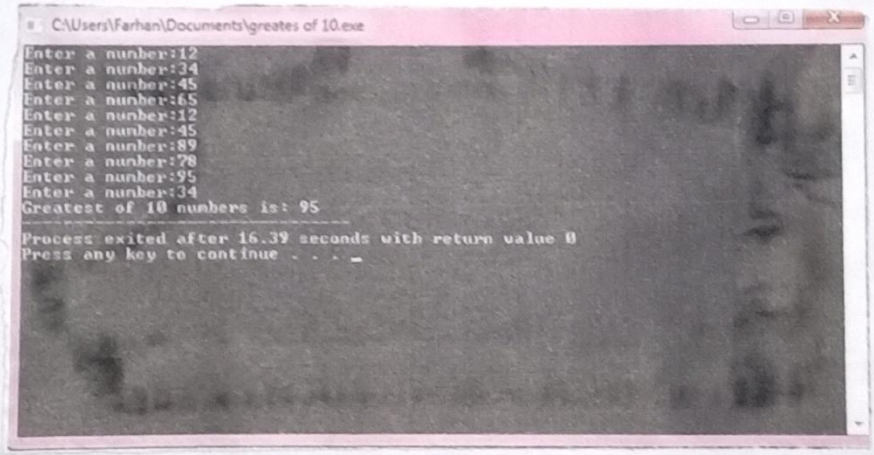
1. Start
2. Declare integer ~~val~~ variable 'max', 'new' and 'i'.
3. Input ten numbers using for loop
4. After each entry check the greatest of available numbers.
5. Print greatest number
6. End.

⇒ Code:

```
#include <stdio.h>
int main()
{   int max, new, i;
```

```
printf("Enter a number");
for (i = 0; i<10; i++)
{       printf("Enter a number:");
        scanf("%d", &new)
        if (i==0)
                max = new;
        else
        { if (new>max)
                max = new
          }
}
printf("The greatest of 10 numbers is %d", max);
}
```

⇒ Output:



⇒ Learning from Experiment:

We learned to use 'for' loop to input ten numbers, instead of creating ten variable for each number.

Program - 3

⇒ **Objective:**

Write a program to find Simple Interest.

⇒ **Introduction:**

The simple interest can be calculated using three parameters, principal amount, rate of interest and duration of loans (in years). These three parameters will be input from the user and the simple interest will be calculated using the standard formula:

$$S.I. = \frac{P \times R \times T}{100}$$

where P = principal amount
R = Rate of interest
T = Duration of loan (in years)
and S.I = Simple interest.

⇒ **Algorithm:**

1. Start
2. Declare integer variable p and float variables r, t, si.
3. Input principal amount (p), rate of interest (r), and time (t).
4. Perform the required operation.

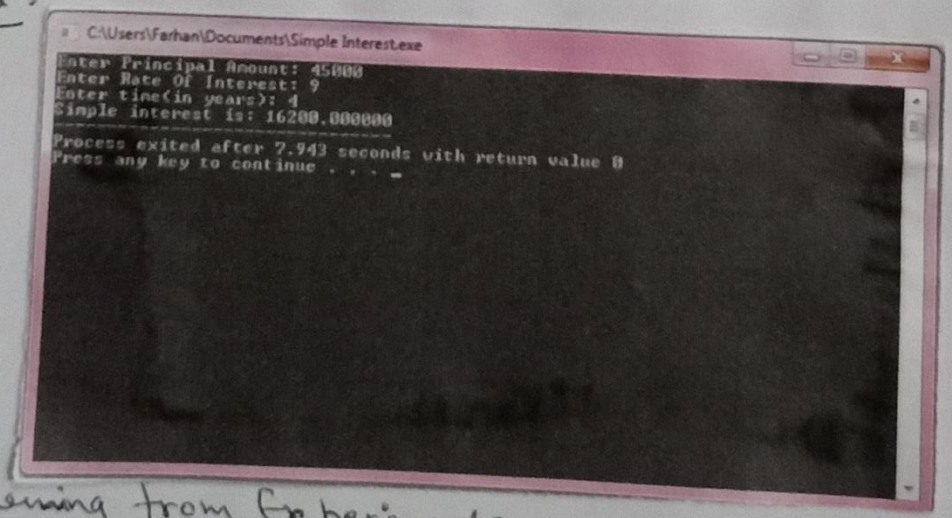
$$si = \frac{p*r*t}{100}$$

5. Print Simple Interest (si).
6. End

=> Code:

```
#include <stdio.h>
int main()
{
    int p;
    float r, t, si;

    printf("Enter Principal amount: ");
    scanf("%d", &p);

    printf("Enter rate of interest: ");
    scanf("%f", &r);

    printf("Enter duration of loan in years: ");
    scanf("%f", &t);

    si = (p*r*t)/100;
    printf("The Simple Interest is: %f", si);

    getch();
    return 0;
}
```

Output:



Learning from Experiment:

Learned formula of simple interest and how to write it in program. Also learned how to do basic operations in program.

Program - 4

⇒ **Objective:**

Write a program to print a triangle of stars.

⇒ **Introduction:**

To print a triangle pyramid using characters, the program is supposed to read the value of number of rows in the character triangle, say n, and then print characters such that there is one character in one row, two in second row and so till n characters in nth row. Also the program will leave spaces before characters in each row, in a dereamental order, i.e. maximum spaces in first row and minimum in last row.

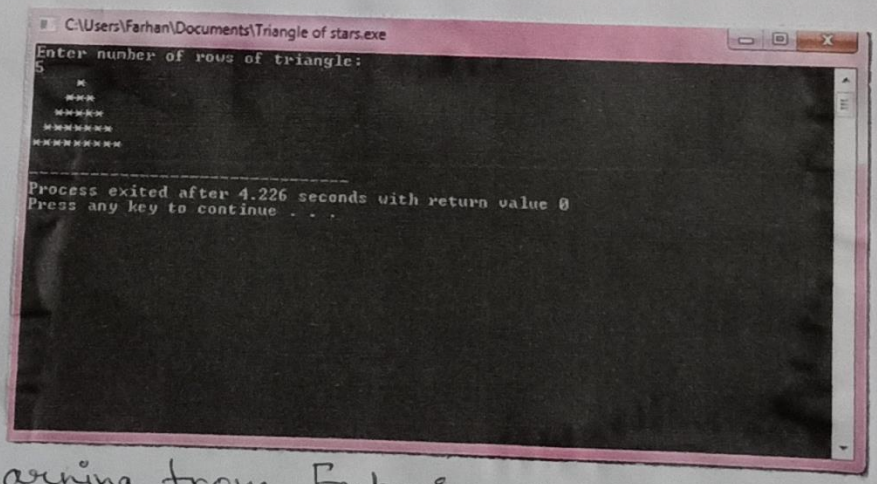
⇒ **Algorithm:**

1. Start
2. Declare integer variable for number of rows. 'space' for no. of spaces.
3. First 'for' loop will control the rows
4. Second for loop will control spaces
5. Third while loop will control printing of star.
6. End.

=> Code:

```
#include <stdio.h>
  void main()
{
  int i, space, n, k=0;

 printf("Enter number of rows:");
 scanf("%d", &n);

 for(i=1; i<=n; i++)
{
    for(space=1; space=n-i; ++space)
      { printf(" "); }
          while (k!=2*i-1)
        { printf ("*");
              ++k; }
          printf("\n"); }
    return 0;
}
```

=> Output:



=> Learning from Experiment:

We have learned how to utilise nested for loop and while loop to print symbols and shapes of our choice

Program-5

⇒ **Objective:**

Write a program to find whether entered number is a prime.

⇒ **Introduction:**

The program will read a value from the user and check for it being a prime or not. Starting from 2, the program will divide the input number with each numb until it is divided with each number till one less than itsel If the remainder in each case remains not equal to zero, the program declares the input as a prime number If the remainder in any case was zero, the input is classified as non-prime.

⇒ **Algorithm:**

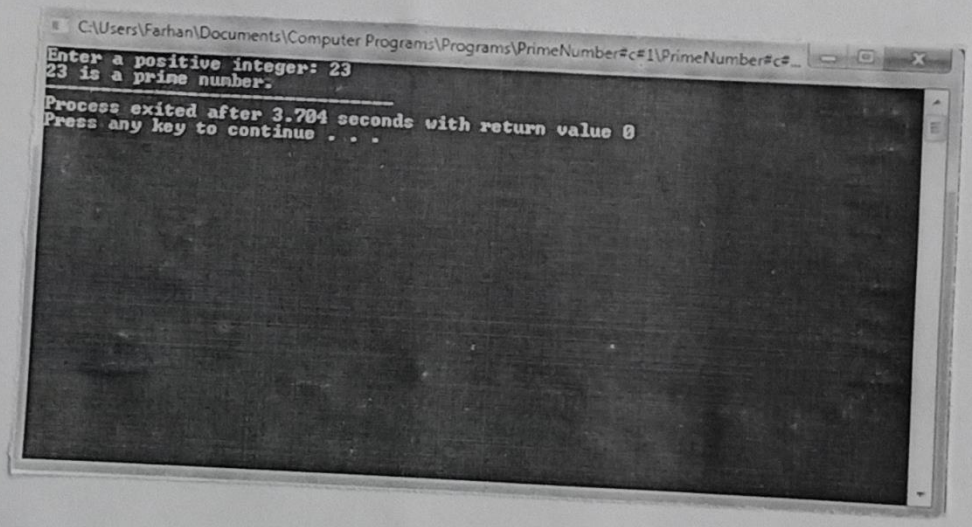
1. Start
2. Declare integer variable x and i
3. Divide x by i, where i is starting from 2 to x-1.
4. If remainder in any case is zero, print number is not prime.
5. If remainder remains non-zero in every case, print number is prime.
6. End.

⇒ **Code:**

```
#include <stdio.h>
int main()
{
    int x, i;
```

```
printf("Enter a number:");
scanf("%d", &x);

for (i=2; i<x-1; i++)
{
    if (x%i==0)
      { printf("%d is not a prime number";
        break;      }
  else
    printf("%d is a prime number", x);
  }
    return 0;
}
```

=) Output:



=) Learning from experiment:

We have learned how to check whether a number is prime or not of which no formula exists in math. We have also utilised use of ✂ break statement.

THANK YOU