DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CO 101

LAB FILE

SUBMITTED TO: ANKRITI KAUSHAL

SUBMITTED BY: UTKARSH PANDEY

CSE 7

(23/CS/446)

INDEX

S NO	OBJECTIVE	DATE	SIGN
1	Write a C program to take input two integers and print their sum.	22/08/2023	
2	Write a C program to take input four integers and print their average.	22/08/2023	
3	Write a C program to implement Relational Operators.	29/08/2023	
4	Write a C program to implement Logical Operators.	29/08/2023	
5	Write a C program to implement Bitwise Operators.	29/08/2023	
6	Write a C program to find the sum of a 3-digit number.	29/08/2023	
7	Write a C program to reverse a 3-digit number.	29/08/2023	
8	Write a program to implement Linear Search Algorithm.	05/9/2023	
9	Write a program to implement Binary Search Algorithm (recursion).	05/9/2023	
10	Write a program to take input a String Using scanf, gets and fget	05/9/2023	
11	Write a Program to Print the following patterns	12/9/23	
12	Write a Program to convert decimal to binary and vice versa.	19/9/23	
13	Write a Program to implement the switch case statement.	19/9/23	
14	Write a Program to generate the Fibonacci sequence using recursion.	19/9/23	
15	Write a Program to create an exponential function.	19/9/23	
16	Write a Program to count the number of vowels in a given string.	17/10/23	
17	Write a Program to check if a given string is a palindrome or not.	17/10/23	
18	Write a Program to string concatenation.	17/10/23	
19	Write a Program to string comparison.	17/10/23	
20	Write a Program to string reverse.	17/10/23	
21	Write a Program to convert a string from lower case to upper case and vice versa.	17/10/23	
22	Program for the addition of two 3 x 3 matrices.	19/10/2023	
23	Program to multiply two 3 x 3 matrices	19/10/2023	
24	Program to find factorial of a number using recursion.	19/10/2023	
25	Write a C program to sort an array using Bubble sort.	19/10/2023	
26	Write a C program to sort an array using selection sort.	19/10/2023	
27	Write a C program to sort an array using insertion sort.	19/10/2023	

EXPERIMENT 1

AIM: Write a C program to take input two integers and print their sum.

THEORY: In this program, the user is asked to enter two integers. These two integers are stored in variables a and b respectively.
Then, these two numbers are added using the + operator and stored in c.

ALGORITHM : Start Procedure

Step 1: PRINT Enter two numbers

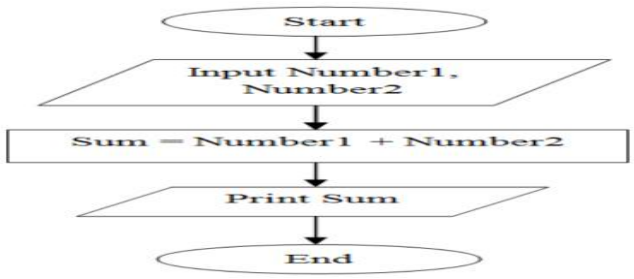
Step 2: READ a , b

Step 3: SET c := a + b

Step 4: PRINT c

End procedure

Flowchart:



CODE : #include<stdio.h>

int main()

{int a,b,c;

printf("Enter two numbers \n");

scanf("%d %d",&a,&b);

c=a+b;

printf("The sum of these numbers is %d",c);}

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p1.c
p1 } ; if ($?) { .\p1 }
Enter two numbers
5
6
The sum of these numbers is 11
```

EXPERIMENT 2

AIM: Write a C program to take input four integers and print their average.

THEORY: In this program, the user is asked to enter four integers. These four integers are stored in variables num1, num2, num3, num4 respectively.
Then, these four numbers are added using the + operator and stored in sum.
Then using '/' g is divided by four and stored in average (of data type float)

ALGORITHM : Start Procedure

Step 1: PRINT Enter four numbers

Step 2: READ num1 ,num2, num3, num4

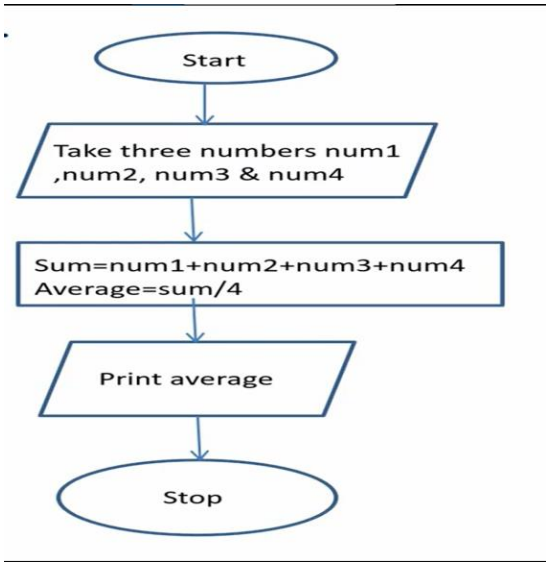
Step 3: SET Sum = num1 +num2+ num3+ num4

Step 4: SET average=Sum/4

Step 5: PRINT average

End procedure

FLOWCHART:



CODE: #include<stdio.h>

int main()

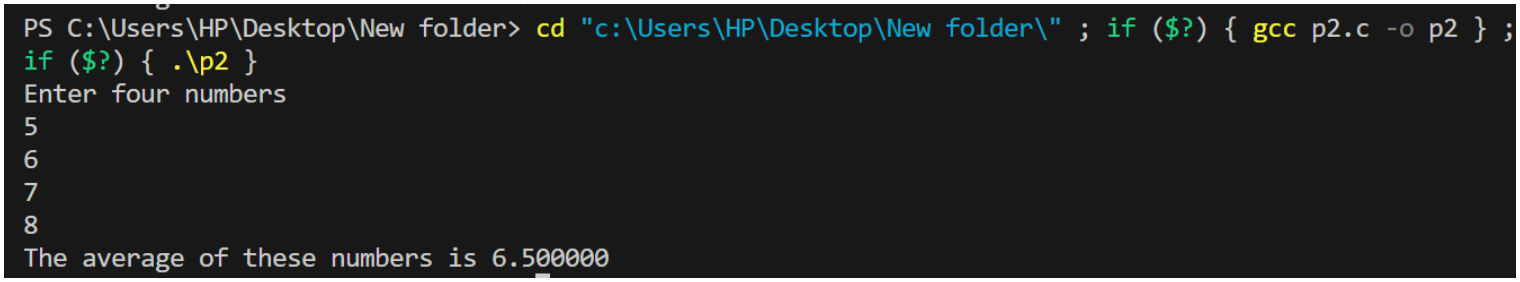
{int num1 ,num2, num3, num4;

float sum,average;

printf("Enter four numbers \n");

```
scanf("%d %d %d %d”,&num1 ,&num2,& num3, &num4);

sum = num1 +num2+ num3+ num4

average=sum/4;

printf("The average of these numbers is %f",average);}
```

OUTPUT:



EXPERIMENT 3

AIM: Write a C program to implement Relational Operators.

THEORY: The relational operators are used to compare two of the available values to understand what relationship the pairs of values share. For example, equal to, greater than, less than, etc. Here is a list of all the relational operators used in the C language:
1)Equal to(==) 2)Not equal to(!=) 3)Less than(<) 4)Greater than(>) 5)Less than or equal to(<=) 6)Greater than or equal to(>=)

ALGORITHM : Start Procedure

Step 1: SET a := 100, b := 200

Step 2: PRINT a = = b

Step 3: PRINT a > b

Step 4: PRINT a < b

Step 5: PRINT a ! = b

Step 6: RETURN 0

End procedure

CODE: #include <stdio.h>

int main()

{ int a=100,b=200;

printf("A equal to b is %d \n",a==b);

printf("A greater than b is %d \n",a>b);

printf("A less than b is %d \n",a<b);

printf("A not equal to b is %d \n",a!=b);}

OUTPUT:

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p3.c -o
p3 } ; if ($?) { .\p3 }
A equal to b is 0
A greater than b is 0
A less than b is 1
A not equal to b is 1
```

EXPERIMENT 4

AIM: Write a C program to implement Logical Operators.

THEORY: The logical operators evaluate the logical expression and return a result. The result is always a Boolean value. A Boolean value determines whether the expression is true or false . There are three logical operators in C programming: logical AND(&&), logical OR(||), and logical NOT (!).

ALGORITHM : Start Procedure

Step 1: PRINT Enter 3 numbers to check relation between them

Step 2: READ a , b , c

Step 3: if a = b and b = c then :

Step 1: PRINT All numbers are equal

Step 4: else :

Step 1: if a = b or b = c then :

Step 1: PRINT Any two numbers are equal

else :

Step 1: PRINT All number are different

End of if else block

End of if else block

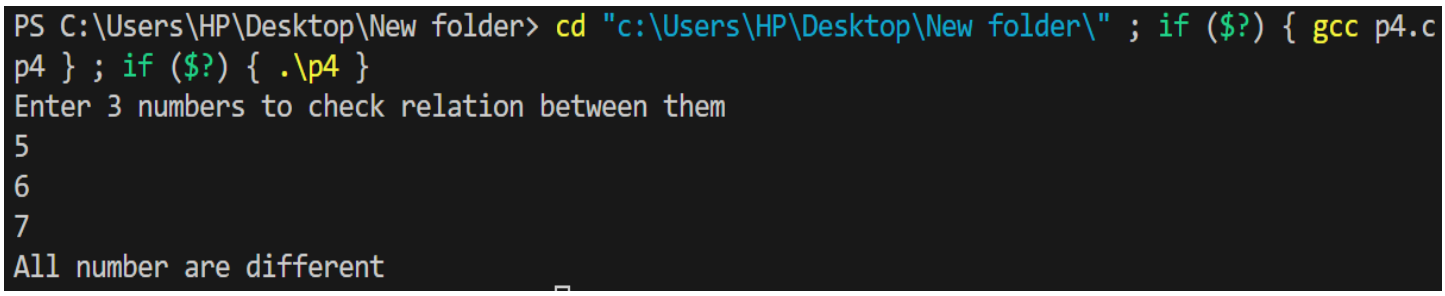
End procedure

CODE: #include <stdio.h>

```
int main()
{
    int a,b,c;
    printf("Enter 3 numbers to check relation between them \n");
    scanf("%d %d %d",&a,&b,&c);
    if (a==b && b==c)
     printf("All numbers are equal");
     else
```

```
    {
    if (a==b || b==c)
    printf("Any two numbers are equal");
    else
    printf("All number are different");}
   return 0;
}
```

OUTPUT:



EXPERIMENT 5

AIM: Write a C program to implement Bitwise Operators.

THEORY: We use the bitwise operators in C language to perform operations on the available data at a bit level. Thus, performing a bitwise operation is also called bit-level programming. It is mainly used in numerical computations for a faster calculation because it consists of two digits – 1 or 0.

Here is a list of the ones used in C:

1) Bitwise OR Operator

2) Bitwise AND Operator

3) Unary Operator (Binary One's complement operator)

4) Bitwise XOR Operator

5) Binary Right Shift Operator

6) Binary Left Shift Operator

ALGORITHM : Start Procedure

Step 1: PRINT Enter the number (A)

Step 2: READ a

Step 3: PRINT Enter the number (B)

Step 4: READ b

Step 5: PRINT (a < < 1)

Step 6: PRINT (a > > 1)

Step 7: PRINT (a < < 5)

Step 8: PRINT (a > > 5)

Step 9: PRINT (a | b)

Step 10: PRINT (a & b)

Step 11: PRINT (a ^ b)

End procedure

CODE: #include <stdio.h>

int main()

```
{int a,b;
        printf("Enter the number(A)");
        scanf("%d",&a);
        printf("Enter the number(B)");
        scanf("%d",&b);
         printf("left shift of A by 1 is %d \n ",(a << 1));
         printf("Right shift of A by 1 is %d \n",(a >> 1));
         printf("left shift of A by 5 is %d \n",(a << 5));
         printf("Right shift of A by 5 is %d \n",(a >> 5));
         printf("A or B is %d \n",(a|b));
         printf("A and B is %d \n",(a&b));
         printf("A XOR B is %d \n",(a^b));}
```

OUTPUT:

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc tempCode
RunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter the number(A)6
Enter the number(B)7
left shift of A by 1 is 12
 Right shift of A by 1 is 3
left shift of A by 5 is 192
Right shift of A by 5 is 0
A or B is 7
A and B is 6
A XOR B is 1
```

EXPERIMENT 6

AIM: Write a C program to find the sum of a 3-digit number.

THEORY: In programming, a loop is used to repeat a block of code until the specified condition is met.

ALGORITHM : Start Procedure

Step 1: PRINT Enter the numbers

Step 2: SET s := 0

Step 3: READ i

Step 4: Repeat step 1 to 3 while i ! = 0 ;

Step 1: SET g := i % 10

Step 2: SET s := s + g

Step 3: SET i := i/10

End of for block

Step 4: PRINT s

End procedure

CODE:

```
#include<stdio.h>
int main()
{int g,i,s=0;
printf("Enter the numbers");
scanf("%d",&i);
for (i;i!=0;i=i/10)
{g=i%10;
s=s+g;}
printf("The sum of these numbers is %d",s);}
```

OUTPUT:

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p6.c -o
p6 } ; if ($?) { .\p6 }
Enter the numbers634
The sum of these numbers is 13
```

EXPERIMENT 7

AIM: Write a C program reverse a 3-digit number.

THEORY: : In programming, a loop is used to repeat a block of code until the specified condition is met.

ALGORITHM : Start Procedure

Step 1: PRINT Enter the number

Step 2: SET s := 0

Step 3: READ i

Step 4: Repeat step 1 to 3 while i ! = 0 ;

Step 1: SET g := i % 10

Step 2: SET s := s * 10 + g

Step 3: SET i := i/10

End of for block

Step 4: PRINT s

End procedure

CODE: #include<stdio.h>

```
int main()
{int g,i,s=0;
printf("Enter the number");
scanf("%d",&i);
for (i;i!=0;i=i/10)
{g=i%10;
s=s*10+g;}
printf("The reverse of the number is %d",s);}
```

OUTPUT:

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p7.c -o
p7 } ; if ($?) { .\p7 }
Enter the number894
The reverse of the number is 498
```

EXPERIMENT 8

AIM: Write a program to implement Linear Search Algorithm.

THEORY: Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set. Its time complexity is O(n).

ALGORITHM : Start Procedure

Step 1: PRINT Enter no of elements

Step 2: READ n

Step 3: SET c := 0

Step 4: Repeat step 1 to 3 while i < n ;

Step 1: PRINT i + 1

Step 2: READ arr [i]

Step 3: SET i := i + 1

Step 4: PRINT Enter key :

Step 5: READ x

Step 6: Repeat step 1 to 2 while i < n ;

Step 1: if x = arr [i] then :

Step 1: SET c := 1

Step 2: PRINT i + 1

End of if block

Step 2: SET i := i + 1

End of for block

Step 6: if c = 0 then :

Step 1: PRINT Key not found

End of if block

End procedure

CODE

```
#include <stdio.h>

int main() {

   int n;

   printf("Enter no of elements");

   scanf("%d", &n);

   int arr[n];

   int x, c = 0, i;

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

      printf("Enter element no %d: ",i+1);

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

   }

   printf("Enter key: ");

   scanf("%d", &x);

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

      if (x == arr[i]) {

         c = 1;

         printf("The key is at %d th position", i+1);

      }

   }

   if (c == 0) {

      printf("Key not found");

   }}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p8.c -o
p8 } ; if ($?) { .\p8 }
Enter no of elements5
Enter element no 1: 1
Enter element no 2: 2
Enter element no 3: 3
Enter element no 4: 4
Enter element no 5: 5
Enter key: 3
The key is at 3 th position
```

EXPERIMENT 9

AIM: Write a program to implement Binary Search Algorithm (recursion).

THEORY: Binary Search is defined as 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 : Start procedure

Define function binarySearch(int arr [ ] , int l , int r , int x):-

Step 1: if r > = l then :

Step 1: SET mid := l + (r - l) / 2

Step 2: if arr [mid] = x then :

Step 1: RETURN mid

End of if block

Step 3: if arr [mid] > x then :

Step 1: RETURN binarySearch (arr , l , mid - 1 , x)

End of if block

Step 4: RETURN binarySearch (arr , mid + 1 , r , x)

End of if block

Step 2: RETURN - 1

End function definition

Start Procedure

Step 1: Declare an array

Step 1: PRINT Enter element

Step 2: READ x

Step 3:Set result= binarySearch(arr, 0, 7, x);

Step 3: if result = - 1 then :

Step 1: PRINT Element is not present in array

: else :

Step 1: PRINT result

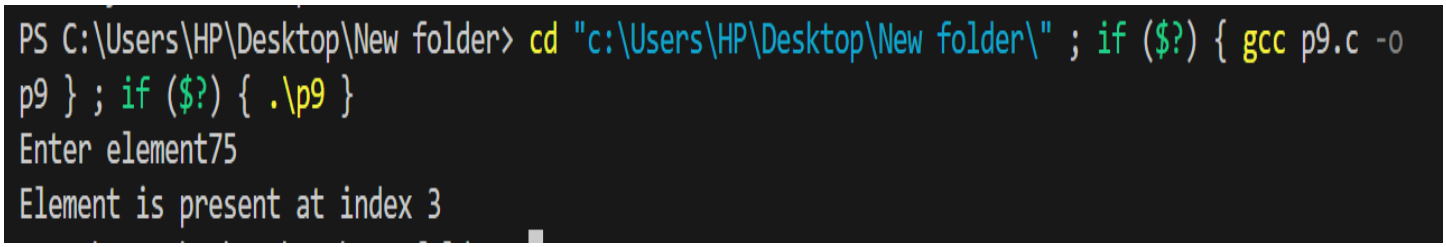
End of if else block

End procedure

CODE #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()
{  int arr[8] = {14,18,28,75,49,75,84,21};
   int x;
   printf("Enter element");
   scanf("%d",&x);
   int result = binarySearch(arr, 0, 7, x);
   (result == -1)? printf("Element is not present in array"): printf("Element is present at index %d", result);
}
```

OUTPUT :



EXPERIMENT 10

AIM: Write a program to take input a String Using scanf, gets and fget

THEORY: Scanf() is a library function in C. It reads standard input from stdin

The C library function char *fgets(char *str, int n, FILE *stream) reads a line from the specified stream and stores it into the string pointed to by str. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.

Gets() reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached.

ALGORITHM Start Procedure

Step 1: Declare character array(cr,string,arr)

Step 2: PRINT Enter the String :

Step 3: READ cr(using fgets)

Step 4: PRINT cr

Step 5: PRINT Enter the String :

Step 6: READ string(using gets)

Step 7: PRINT string

Step 8: PRINT Enter string :

Step 9: READ arr(using scanf)

Step 10: PRINT arr

End procedure

CODE #include <stdio.h>

```
int main()
{ char cr[15];
printf("Enter the String: ");
   fgets(cr,15, stdin);
   printf("Your entred string (using fgets) is  %s", cr);
char string[10];
```

```
printf("Enter the String: ");
   gets(string);
   printf("Your entred string (using gets) is %s\n",string);
char arr[3]="";
  printf("Enter string: ");
   scanf("%s", &arr);
   printf("\n Your entred string (using scanf) is %s ",arr);
   return 0;
}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p10.c -o
 p10 } ; if ($?) { .\p10 }
Enter the String: HELLO
Your entred string (using fgets) is  HELLO
Enter the String: GREEK
Your entred string (using gets) is GREEK
Enter string: ALPHA
```

EXPERIMENT 11

AIM: Write a Program to Print the following patterns

THEORY: We use two nested loops for rows and column respectively and use if else statement as per requirement.

ALGORITHM :

Start Procedure

Step 1: Repeat step 1 to 2 while i < = 6 ;

Step 1: Repeat step 1 to 3 while j < = 6 ;

Step 1: if j < = i then :

Step 1: PRINT *

else :

Step 1: PRINT

End of if else block

Step 3: SET j := j + 1

End of for block

Step 1: PRINT \ n

Step 2: SET i := i + 1

End of for block

Step 2: Repeat step 1 to 2 while i < = 6 ;

Step 1: Repeat step 1 to 3 while j < = 6 ;

Step 1: if j < = 7 - i then :

Step 1: PRINT *

else :

Step 1: PRINT

End of if else block

Step 3: SET j := j + 1

```
        End of for block
        Step 1: PRINT \ n
        Step 2: SET i := i + 1
End of for block
Step 3: Repeat step 1 to 2 while i < = 6 ;
        Step 1: Repeat step 1 to 3 while j < = 6 ;
                Step 1: if i = 1 or j = 1 or j = 7 - i then :
                        Step 1: PRINT *
                        else :
                        Step 1: PRINT
                End of if  else block
                Step 3: SET j := j + 1
        End of for block
        Step 1: PRINT \ n
        Step 2: SET i := i + 1
End of for block
Step 4: SET k := 0
Step 5: Repeat step 1 to 2 while i < = 6 ;
        Step 1: Repeat step 1 to 3 while j < = 11 ;
                Step 1: if j > = 7 - i and j < = 5 + i and k = 0 then :
                        Step 1: PRINT *
                        Step 2: SET k := 1
                       else :
                        Step 1: PRINT
                        Step 2: SET k := 0
                End of if  else block
                Step 3: SET j := j + 1
```

```
        End of for block
        Step 1: PRINT \ n
        Step 2: SET i := i + 1
End of for block
Step 6: SET k := 0
Step 7: Repeat step 1 to 2 while i < = 6 ;
        Step 1: Repeat step 1 to 3 while j < = 11 ;
                Step 1: if j > = i and j < = 12 - i and k = 0 then :
                        Step 1: PRINT *
                        Step 2: SET k := 1
                         else :
                        Step 1: PRINT
                        Step 2: SET k := 0
                End of else block
                Step 3: SET j := j + 1
        End of for block
        Step 1: PRINT \ n
        Step 2: SET i := i + 1
End of for block
Step 7: Repeat step 1 to 2 while i < = 6 ;
        Step 1: Repeat step 1 to 3 while j < = 11 ;
                Step 1: if j = 7 - i or j = 5 + i then :
                        Step 1: PRINT *
                        else :
                        Step 1: PRINT
                End of else block
                Step 3: SET j := j + 1
```

End of for block

Step 1: PRINT \ n

Step 2: SET i := i + 1

End of for block

End procedure

CODE : #include <stdio.h>

```
int main() {
   for (int i = 1; i <= 6; i++) {
      for (int j = 1; j <= 6; j++) {
         if (j<=i)
            printf("*");
         else
            printf(" ");
      }
      printf("\n");
   }
printf("\n");
   for (int i = 1; i <= 6; i++) {
      for (int j = 1; j <= 6; j++) {
         if (j<=7-i)
            printf("*");
         else
            printf(" ");
      }
      printf("\n");
   }
   printf("\n");
```

```
 for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 6;j++) {
        if (i==1 || j==1 || j==7-i)
            printf("*");
        else
            printf(" ");}
    printf("\n");}
 printf("\n");
 int k=0;
  for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 11;j++) {
        if (j>=7-i && j<=5+i && k==0)
            {printf("*");
            k=1;}
        else
            {printf(" ");
            k=0;}
    }
    printf("\n");
 }
printf("\n");  k=0;
for (int i = 1; i <= 6; i++) {
    for (int j = 1; j <= 11;j++) {
        if (j>=i && j<=12-i && k==0)
            {printf("*");
            k=1;}
```

```
        else
            {printf(" ");
            k=0;}
      }
      printf("\n");}
   printf("\n");
    for (int i = 1; i <= 6; i++) {
      for (int j = 1; j <= 11;j++) {
        if (j==7-i|| j==5+i || (i==6 && (j==1 || j==3 || j==5 ||j==7||j==9||j==11)))
            {printf("*");}
        else
            {printf(" ");}   }
      printf("\n");
}
}
```

OUTPUT :

EXPERIMENT 12

AIM: Write a Program to convert decimal to binary and vice versa.

THEORY: The number system that represents a number in terms of 0 to 9 digits is a decimal number system. A decimal number system has ten digits, i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The base of a number in this system is 10. In the decimal number system, a number is expressed in terms of powers of 10. A number system that expresses a number in terms of 0 and 1 digits is a binary number system. A binary system has only two digits, i.e., 0 and 1. The base of a number in this system is 2. In a binary number system, a number is expressed in terms of powers of 2. For example, a decimal number 26 is expressed as (11010)2 in a binary system

ALGORITHM :

Start Procedure

Step 1: PRINT Enter decimal number :

Step 2: READ a

Step 3: SET i := 0

Step 4: Repeat step 1 to 3 while a ! = 0

    Step 1: SET arr [i] := a % 2

    Step 2: SET a := a / 2

    Step 3: SET i := i + 1

End of while block

Step 4: PRINT Your binary equivalent is :

Step 5: Repeat step 1 to 2 while j < i ;

    Step 1: PRINT arr [i - j - 1]

    Step 2: SET j := j + 1

End of for block

Step 5: PRINT Enter binary number :

Step 6: READ b

Step 7: SET sum := 0

Step 8: SET k := 0

Step 9: Repeat step 1 to 4 while b ! = 0

Step 1: SET c := b % 10

Step 2: if c = 1 then :

Step 1: SET sum := sum + power (2 , k)

End of if block

Step 3: SET b := b / 10

Step 4: SET k := k + 1

End of while block

Step 9: PRINT sum

End procedure

CODE : #include <stdio.h>

```
int power(int x, int y) {
   int r = 1;
   while (y != 0) {
      y--;
      r = r * x;
   } return r;
}
int main() {
   int a;
   int arr[100];
   printf("Enter decimal number: ");
   scanf("%d", &a);
   int i = 0;
   while (a != 0) {
      arr[i] = a % 2;
      a = a / 2;
      i++;
```

```
    }
    printf("Your binary equivalent is: ");
    for (int j = 0; j < i; j++) {
        printf("%d", arr[i - j - 1]);
    }
    int b;
    printf("\nEnter binary number: ");
    scanf("%d", &b);
    int sum = 0;
    int k = 0;
    while (b != 0) {
        int c = b % 10;
        if (c == 1) {
            sum = sum + power(2, k);
        }
        b = b / 10;
        k++;
    }
    printf("\nThe binary equivalent is: %d", sum);
}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p12.c -o p12 }
 ; if ($?) { .\p12 }
Enter decimal number: 34
Your binary equivalent is: 100010
Enter binary number: 101011

The binary equivalent is: 43
```

EXPERIMENT 13

AIM: Write a Program to implement the switch case statement.

THEORY: Switch case statement evaluates a given expression and based on the evaluated value(matching a certain condition), it executes the statements associated with it. Basically, it is used to perform different actions based on different conditions(cases).

ALGORITHM : Start procedure

Step 1: PRINT Enter first number =

Step 2: READ num1

Step 3: PRINT Enter second number =

Step 4: READ num2

Step 5: PRINT Choose operator to perform operations =

Step 6: READ ch

Step 7: SET result := 0

Step 8:Check ch and perform accordingly and set result as per input.

Step 9: PRINT num1 , ch , num2 , result

End procedure

CODE

```
#include<stdio.h>
int main ()
{int num1, num2;
 float result;
 char ch;
 printf ("Enter first number = ");
 scanf ("%d", &num1);
 printf ("Enter second number = ");
 scanf ("%d", &num2);
 printf ("Choose operator to perform operations = ");
 scanf (" %c", &ch);
```

```
  result = 0;
  switch (ch)
   {case '+':
     result = num1 + num2;
     break;
   case '-':
     result = num1 - num2;
     break;
   case '*':
     result = num1 * num2;
     break;
   case '/':
     result = num1 / num2;
     break;
   default:
     printf ("Invalid operation\n");
   }
  printf ("Result: %d %c %d = %f\n", num1, ch, num2, result);
  return 0;
}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p13.c -o p13 }
 ; if ($?) { .\p13 }
Enter first number = 15
Enter second number = 5
Choose operator to perform operations = *
Result: 15 * 5 = 75.000000
```

EXPERIMENT 14

AIM: Write a Program to generate the Fibonacci sequence using recursion.

THEORY: In mathematics, the Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones. The C programming language allows any of its functions to call itself multiple times in a program. Here, any function that happens to call itself again and again (directly or indirectly), unless the program satisfies some specific condition/subtask is called a recursive function.

ALGORITHM : Start procedure

Define function fib(int i):-

Step 1: if i = 1 then :

Step 1: RETURN 0

Else if :

if i = 3 or i = 2 then :

Step 1: RETURN 1

else :

RETURN fib (i - 1) + fib (i - 2)

End of if else block

End procedure

Step 4: PRINT Enter no of terms

Step 5: READ n

Step 6: Repeat step 1 to 2 while i < = n ;

Step 1: PRINT fib (i)

Step 2: SET i := i + 1

End of for block

End procedure

CODE #include <stdio.h>

int main()

{ int x,i,n;

```
 int fib(int i)

 {

if (i==1)

return 0;

else if (i==3 ||i==2)

{

  return 1;

}

else

return fib(i-1)+fib(i-2);

}

printf("Enter no of terms");

scanf("%d",&n);

for(i=1;i<=n;i++)

{printf("%d ",fib(i));}

}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc P14.c -o P14
 ; if ($?) { .\P14 }
Enter no of terms15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

EXPERIMENT 15

AIM: Write a Program to create an exponential function.

THEORY: An exponential function is a Mathematical function in the form f (x) = ax, where “x” is a variable and “a” is a constant which is called the base of the function and it should be greater than 0.

ALGORITHM : Start procedure

Step 1 Define function power(int x , int y):-

Step 1: SET r := 1

Step 2: Repeat step 1 to 2 while y ! = 0

Step 1: SET y := y - 1

Step 2: SET r := r * x

End of while block

Step 3: RETURN r

End of function.

Step 2: PRINT Enter power & Enter power

Step 3: READ x and y respectively

Step 4: PRINT the result is power(x,y).

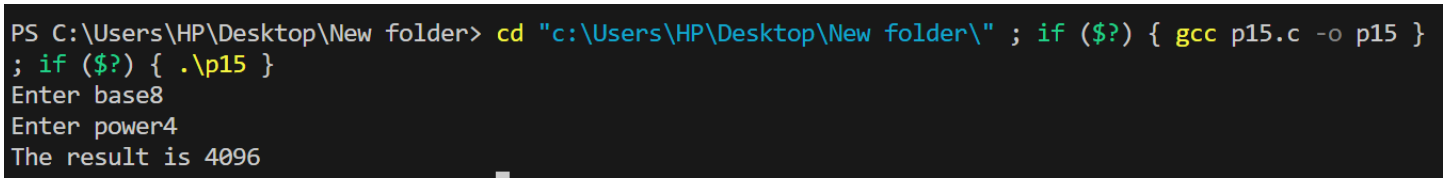
End procedure

CODE #include <stdio.h>

```
int main()
{ int power(int x,int y)
 {int r=1;
   while (y!=0)
 {y--;
  r=r*x;
 } return r; }
int x,y;
```

```
printf("Enter base");

scanf("%d",&x);

printf("Enter power");

scanf("%d",&y);

{printf("The result is %d ",power(x,y));}

}
```

OUTPUT :



EXPERIMENT 16

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

THEORY: In English, the vowels are a, e, i, o, and u.

The strlen() function takes a string as an argument and returns its length. The returned value is of type size_t (an unsigned integer type).It is defined in the <string.h> header file.

ALGORITHM :

Start Procedure

Step 1: PRINT Enter String

Step 2: SET h := strlen (a)

Step 3: SET g := 0

Step 4: Repeat step 1 to 2 while i < h ;

Step 1: if a [i] = ' A ' or a [i] = ' B ' or a [i] = ' I ' or a [i] = ' O ' or a [i] = ' U ' then :

Step 1: SET g := g + 1

End of if block

Step 2: SET i := i + 1

End of for block

Step 4: PRINT g

End procedure

CODE

```
#include <stdio.h>

#include <string.h>

int main()

{

   char a[10];

   printf("Enter String  ");

   gets(a);

int h=strlen(a);
```

```
int g=0;
   for(int i=0;i<h;i++)
   {
      if (a[i]=='A' || a[i]=='E' || a[i]=='I' || a[i]=='O' || a[i]=='U' )
      g++;
   }
   printf("%d",g);
}
```

OUTPUT

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p16.c -o p16 }
; if ($?) { .\p16 }
Enter String  ALPHABET
3
```

EXPERIMENT 17

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

THEORY: A string is said to be palindrome if it remains the same on reading from both ends. It means that when you reverse a given string, it should be the same as the original string. For instance, the string 'level' is a palindrome because it remains the same when you read it from the beginning to the end and vice versa. However, the string 'Simplilearn' is not a palindrome as the reverse of the string is 'nraelilpmis,' which is not the same.

ALGORITHM : Start procedure

Step 1: PRINT Enter string

Step 2: Gets string str

Step 3: SET l := 0

Step 4: SET h := strlen (str) - 1

Step 5: SET g := 0

Step 5: Repeat step 1 to 1 while h > l

Step 1: if str [l + +] ! = str [h - -] then :

Step 1: PRINT str is not palindrome.

Step 2: SET g := g + 1

Step 3: Break the loop

End of if block

End of while block

Step 5: if g = 0 then :

Step 1: PRINT str is a palindrome.

End of if block

End procedure

CODE : #include <stdio.h>

#include <string.h>

int main()

```
{char str[] = { "ABCDE" };
printf("Enter string ");
   gets(str);
   int l = 0;
   int h = strlen(str) - 1;
   int g=0;
   while (h > l) {
      if (str[l++] != str[h--]) {
         printf("%s is not a palindrome\n", str);
         g++;
         break;
      }
   }
   if(g==0)
   printf("%s is a palindrome\n", str);
}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p17.c -o p17 }
; if ($?) { .\p17 }
Enter string ABCBA
ABCBA is a palindrome
```

EXPERIMENT 18

AIM: Write a Program to string concatenation.

THEORY: The strcat() function which defined under the string. h header file in c concatenates two strings and returns a string in which the two strings are appended. The function takes the source and destination strings as parameters and returns the destination string where the destination and source strings are appended.

ALGORITHM : Start Procedure

Step 1: Intialized 2 strings

Step 2: Gets two strings

Step 3: Adds the two strings using strcat function.

Step 4:Print the output

End procedure

CODE #include <stdio.h>

#include <string.h>

int main()

{char a[]="ABC";

char b[]="ABC";

int g=0;

printf("Enter two strings: ");

gets(a);

gets(b);

printf("The concatenation of these strings are %s",strcat(a,b));

}

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { g++ P18.C -o P18 }
; if ($?) { .\P18 }
Enter two strings: ABC
DEF
The concatenation of these strings are ABCDEF
```

EXPERIMENT 19

AIM: Write a Program to string comparison.

THEORY: C strcmp() is a built-in library function that is used for string comparison. This function takes two strings (array of characters) as arguments**,** compares these two strings lexicographically, and then returns 0,1, or -1 as the result. It is defined inside <string.h> header file with its prototype as follows:

ALGORITHM : Start Procedure

Step 1: Intialized 2 strings

Step 2: Gets two strings

Step 3: Compare the two strings using strcmp function.

Step 4:Print the output(0 if both are same)

End procedure

CODE: #include <stdio.h>

#include <string.h>

int main()

{char a[]="ABC";

char b[]="ABC";

int g=0;

gets(a);

gets(b);

printf("%d",strcmp(a,b));

}

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc P19.c -o P19 }
; if ($?) { .\P19 }
ABC
ABC
0
```

EXPERIMENT 20

AIM: Write a Program to string reverse.

THEORY: The strrev() function is a built-in function in C and is defined in string.h header file. The strrev() function is used to reverse the given string.

Syntax:char *strrev(char *str);

Returns: This function doesn't return anything but the reversed string is stored in the same string.

ALGORITHM : Start procedure

Define function revstr(char str1):-

Step 1: SET len := strlen (str1)

Step 2: Repeat step 1 to 4 while i < len / 2 ;

Step 1: SET temp := str1 [i]

Step 2: SET str1 [i] := str1 [len - i - 1]

Step 3: SET str1 [len - i - 1] := temp

Step 4: SET i := i + 1

End of for block

End of function definition

Start Procedure

Step 1: PRINT Enter the string :

Step2:Gets str

Step 3: Sets str=revstr(str)

Step 4: PRINT str

End procedure

CODE #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()
    {  char str[50];
        printf (" Enter the string: ");
        gets(str);
        revstr(str);
        printf (" After reversing the string: %s", str);
    }
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc P20.c -o P20 }
; if ($?) { .\P20 }
 Enter the string: COMPUTER
 After reversing the string: RETUPMOC
```

EXPERIMENT 21

AIM: Program to convert a string from lower case to upper case.

THEORY: ASCII stands for American Standard Code for Information Interchange. Below is the ASCII character table, including descriptions of the first 32 characters. ASCII was originally designed for use with teletypes, and so the descriptions are somewhat obscure and their use is frequently not as intended .The different between ASCII value of a alphabet in upper case and lower case is 32

ALGORITHM : Start procedure

Step 1: PRINT Enter a string :

Step 2: Repeat step 1 to 2 while s [i] ! = ' 0 ' ;

Step 1: if s [i] > = ' a ' and s [i] < = ' z ' then :

Step 1: SET s [i] := s [i] - 32

End of if block

Step 2: SET i := i + 1

End of for block

Step 2: PRINT s

Step 3: PRINT Enter a string :

Step 4: Repeat step 1 to 2 while s [i] ! = ' \ 0 ' ;

Step 1: if s [i] > = ' A ' and s [i] < = ' Z ' then :

Step 1: SET s [i] := s [i] + 32

End of if block

Step 2: SET i := i + 1

End of for block

Step 4: PRINT s

End procedure

CODE: #include <stdio.h>

#include <string.h>

int main() {

```
  char s[100];
  int i;
  printf("\nEnter a string : ");
  gets(s);
  for (i = 0; s[i]!='\0'; i++) {
    if(s[i] >= 'a' && s[i] <= 'z') {
      s[i] = s[i] - 32;
    }
  }
  printf("\nString in Upper Case = %s", s);
   printf("\nEnter a string : ");
  gets(s);
  for (i = 0; s[i]!='\0'; i++) {
    if(s[i] >= 'A' && s[i] <= 'Z') {
      s[i] = s[i] + 32;
    }
  }
  printf("\nString in Lower Case = %s", s);
}
```

OUTPUT

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p21.c -o p21 }
; if ($?) { .\p21 }

Enter a string : alpha

String in Upper Case = ALPHA
Enter a string : QWERTY

String in Lower Case = qwerty
```

EXPERIMENT 22

AIM: Program for the addition of two 3 x 3 matrices.

THEORY: Addition of matrix is the basic operation performed, to add two or more matrices. Matrix addition is possible only if the order of the given matrices are the same

$$A + B = [a_{ij}]_{mxn} + [b_{ij}]_{mxn} = [a_{ij} + b_{ij}]_{mxn}$$

ALGORITHM : Start procedure

Step 1: Initialize two matrix

Step 5: Repeat step 1 to 1 while i < 3 ;

Step 1: Repeat step 1 to 2 while j < 3 ;

Step 1: SET r [i] [j] := a1 [i] [j] + a2 [i] [j]

Step 2: SET j := j + 1

End of for block

Step 1: SET i := i + 1

End of for block

Step 2: PRINT The resultant matrix is

Step 3: Repeat step 1 to 2 while i < 3 ;

Step 1: Repeat step 1 to 2 while j < 3 ;

Step 1: PRINT r [i] [j]

Step 2: SET j := j + 1

End of for block

Step 1: PRINT \ n

Step 2: SET i := i + 1

End of for block

End procedure

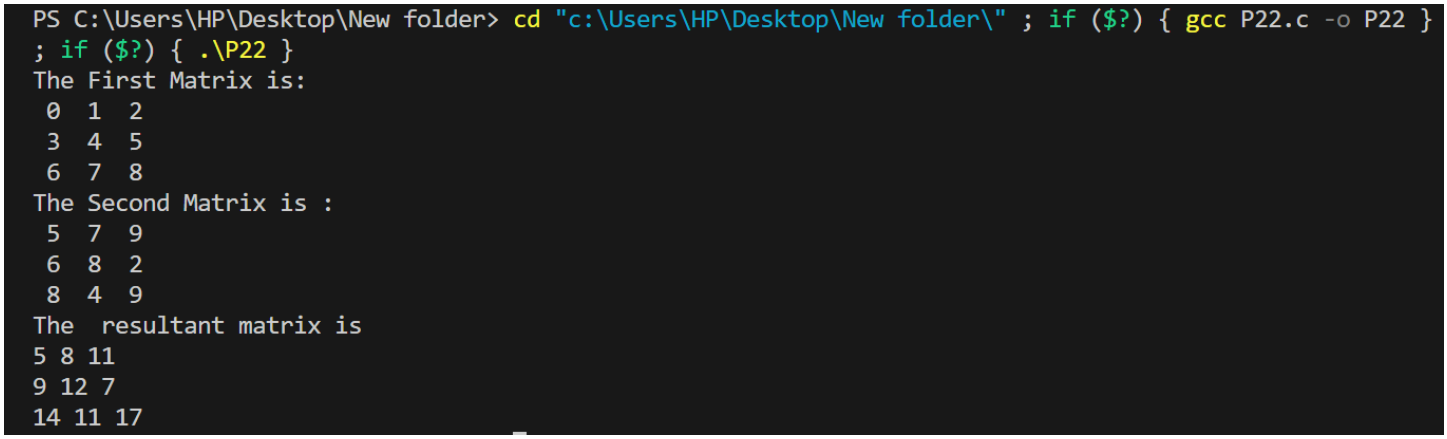
CODE:#include <stdio.h>

int main() {

```
    int a1[3][3]={{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
    int a2[3][3]={{5,7,9}, {6,8,2}, {8,4,9}};
    int r[3][3];
    int i,j;
for(int i = 0; i <3 ; i++){
    for(int j = 0; j <3; j++){
        r[i][j] = a1[i][j] + a2[i][j];
    }
}
printf("The First Matrix is: \n");
  for (i = 0; i < 3; i++) {
     for (j = 0; j < 3; j++) {
        printf(" %d ", a1[i][j]);
     }
     printf("\n");
  }
  printf("The Second Matrix is : \n");
  for (i = 0; i < 3; i++) {
     for (j = 0; j < 3; j++) {
        printf(" %d ", a2[i][j]);
     }
     printf("\n");
  }
printf("The  resultant matrix is \n");
for (int i = 0; i < 3; i++) {
       for (int j = 0; j < 3; j++) {
          printf("%d ", r[i][j]);
```

```
    }

    printf("\n");

  } }
```

OUTPUT :



EXPERIMENT 23

AIM: Program for the multiplication of two 3 x 3 matrices

THEORY: Consider matrix A which is a × b matrix and matrix B, which is a b ×c matrix.

Then, matrix C = AB is defined as the A × B matrix.

An element in matrix C, C_{xy} is defined as:

$$C_{xy} = A_{x1}B_{y1} + \ldots . + A_{xb}B_{by} = \sum_{k=1}^{b} A_{xk}B_{ky}$$

For x = 1..... a and y= 1.....c

ALGORITHM : Start Procedure

Step 1: Initialize two matrix

Step 2: SET sum := 0

Step 7: Repeat step 1 to 1 while i < = 2 ;

Step 1: Repeat step 1 to 3 while j < = 2 ;

Step 1: SET sum := 0

Step 2: Repeat step 1 to 2 while k < = 2 ;

Step 1: SET sum := sum + a [i] [k] * b [k] [j]

Step 2: SET k := k + 1

End of for block

Step 2: SET c [i] [j] := sum

Step 3: SET j := j + 1

End of for block

Step 1: SET i := i + 1

End of for block

Step 8: PRINT Multiplication Of Two Matrices : \ n

Step 9: Repeat step 1 to 2 while i < 3 ;

Step 1: Repeat step 1 to 2 while j < 3 ;

Step 1: PRINT c [i] [j]

Step 2: SET j := j + 1

End of for block

Step 1: PRINT \ n

Step 2: SET i := i + 1

End of for block

End procedure

CODE : #include<stdio.h>

```
int main() {
    int a[3][3]={{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
   int b[3][3]={{5,7,9}, {6,8,2}, {8,4,9}};
    int c[10][10], i, j, k;
  int sum = 0;
  printf("The First Matrix is: \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", a[i][j]);
    }
    printf("\n");
  }
  printf("The Second Matrix is : \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", b[i][j]);
```

```
    }
    printf("\n");
  }
for (i = 0; i <= 2; i++) {
    for (j = 0; j <= 2; j++) {
      sum = 0;
      for (k = 0; k <= 2; k++) {
        sum = sum + a[i][k] * b[k][j];
      }
      c[i][j] = sum;
    }
  }
  printf("\nMultiplication Of Two Matrices : \n");
  for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
      printf(" %d ", c[i][j]);
    }
    printf("\n");
  }
}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p23.c -o p23 }
; if ($?) { .\p23 }
The First Matrix is:
 0  1  2
 3  4  5
 6  7  8
The Second Matrix is :
 5  7  9
 6  8  2
 8  4  9

Multiplication Of Two Matrices :
 22  16  20
 79  73  80
 136  130  140
```

EXPERIMENT 24

AIM: Program to find factorial of a number using recursion.

THEORY: The Factorial of a whole number 'n' is defined as the product of that number with every whole number less than or equal to 'n' till 1.
The recursion process in C refers to the process in which the program repeats a certain section of code in a similar way. Thus, in the programming languages, when the program allows the user to call any function inside the very same function, it is referred to as a recursive call in that function.

ALGORITHM : Start procedure

Define function factorial(int x):-

Step 1: if x = 0 or x = 1 then :

Step 1: RETURN 1

else :

RETURN x * factorial (x - 1)

End function procedure

Step 1: PRINT Enter number :

Step 2: READ a

Step 3: PRINT factorial (a)

End procedure

CODE : #include <stdio.h>

#include <string.h>

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

```
int main()

{int a;

 printf("Enter number:");

 scanf("%d",&a);

 printf("The factorial of the number is %d",factorial(a));

}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p24.c -o p24 }
; if ($?) { .\p24 }
Enter number:9
The factorial of the number is 362880
```

EXPERIMENT 25

AIM: Write a C program to sort an array using Bubble sort.

THEORY: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

ALGORITHM : Start Procedure

Step 1: Declare and initialize array a[10];

Step 2: Repeat step 1 to 2 while i < 9 ;

Step 1: Repeat step 1 to 2 while j < 9 - i ;

Step 1: if a [j] > a [j + 1] then :

Step 1: SET c := a [j]

Step 2: SET a [j] := a [j + 1]

Step 3: SET a [j + 1] := c

End of if block

Step 2: SET j := j + 1

End of for block

Step 1: Repeat step 1 to 2 while k < 10 ;

Step 1: PRINT a [k]

Step 2: SET k := k + 1

End of for block

Step 1: PRINT \ n

Step 2: SET i := i + 1

End of for block

Step 3: Repeat step 1 to 2 while i < 10 ;

Step 1: PRINT a [i]

Step 2: SET i := i + 1

End of for block

End procedure

CODE:

```
#include <stdio.h>

int main() {

   int a[10] = {51, 14, 87, 65, 84, 63, 2, 5, 8, 86};

   int i, j;

  for (i = 0; i < 9; i++) {

      for (j = 0; j < 9 - i; j++) {

         if (a[j] > a[j + 1]) {

            int c;

            c=a[j];

            a[j]=a[j+1];

            a[j+1]=c;

         }

      }

      for (int k = 0; k < 10; k++)

{printf("%d ", a[k]);}

printf("\n");}

for (i = 0; i < 10; i++)

{printf("%d ", a[i]);}

}
```

OUTPUT :

```
PS C:\Users\HP> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p27.c -o p27 } ; if ($?) { .\p27 }
14 51 65 84 63 2 5 8 86 87
14 51 65 63 2 5 8 84 86 87
14 51 63 2 5 8 65 84 86 87
14 51 2 5 8 63 65 84 86 87
14 2 5 8 51 63 65 84 86 87
2 5 8 14 51 63 65 84 86 87
2 5 8 14 51 63 65 84 86 87
2 5 8 14 51 63 65 84 86 87
2 5 8 14 51 63 65 84 86 87
2 5 8 14 51 63 65 84 86 87
```

EXPERIMENT 26

AIM: Write a C program to sort an array using selection sort.

THEORY: Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

ALGORITHM : Start Procedure

Step 1: Declare and initialize array a[10];

Step 2: Repeat step 1 to 6 while i < 9 ;

Step 1: SET min := i

Step 2: Repeat step 1 to 2 while j < 10 ;

Step 1: if a [j] < a [min] then :

Step 1: SET min := j

End of if block

Step 2: SET j := j + 1

End of for block

Step 2: SET c := a [min]

Step 3: SET a [min] := a [i]

Step 4: SET a [i] := c

Step 5: Repeat step 1 to 2 while k < 10 ;

Step 1: PRINT a [k]

Step 2: SET k := k + 1

End of for block

Step 6: SET i := i + 1

End of for block

End procedure

CODE : #include <stdio.h>

int main() {

```
  int a[10] = {51, 14, 87, 65, 84, 63, 2, 5, 8, 86};
  int i, j;
  for (i = 0; i < 9; i++) {
     int min = i;
     for (j = i + 1; j < 10; j++) {
        if (a[j] < a[min]) {
           min = j;
        }
     }
     {int c;
     c=a[min];
     a[min]=a[i];
     a[i]=c;}
for (int k = 0; k < 10; k++) {
        printf(" %d ",a[k]);
     }
printf("\n");
  }}
```

OUTPUT :

```
PS C:\Users\HP\Desktop\New folder> cd "c:\Users\HP\Desktop\New folder\" ; if ($?) { gcc p26.c -o p26 }
; if ($?) { .\p26 }
 2  14  87  65  84  63  51  5  8  86
 2  5  87  65  84  63  51  14  8  86
 2  5  8  65  84  63  51  14  87  86
 2  5  8  14  84  63  51  65  87  86
 2  5  8  14  51  63  84  65  87  86
 2  5  8  14  51  63  84  65  87  86
 2  5  8  14  51  63  65  84  87  86
 2  5  8  14  51  63  65  84  87  86
 2  5  8  14  51  63  65  84  86  87
```

EXPERIMENT 27

AIM: Write a C program to sort an array using insertion sort.

THEORY: Insertion sort is an algorithm used to sort a collection of elements in ascending or descending order. The basic idea behind the algorithm is to divide the list into two parts: a sorted part and an unsorted part.

ALGORITHM : Start procedure

Step 1: Declare and initialize array a[10];

Step 2: Repeat step 1 to 4 while i < 10 ;

Step 1: SET key := arr [i]

Step 2: SET j := i - 1

Step 3: Repeat step 1 to 2 while j > = 0 and arr [j] > key

Step 1: SET arr [j + 1] := arr [j]

Step 2: SET j := j - 1

End of while block

Step 3: SET arr [j + 1] := key

Step 4: Repeat step 1 to 2 while k < 10 ;

Step 1: PRINT arr [k]

Step 2: SET k := k + 1

End of for block

Step 4: SET i := i + 1

End of for block

End procedure

CODE #include <stdio.h>

```
int main() {
  int arr[10]={51,14,87,65,84,63,2,5,8,86};
 {int i, key, j;
  for (i = 1; i < 10; i++)
```

```
  {
     key = arr[i];
     j = i - 1;
     while (j >= 0 && arr[j] > key)
     {arr[j + 1] = arr[j];
        j--;
     }
     arr[j + 1] = key;
     for (int k = 0; k < 10; k++) {
        printf(" %d ",arr[k]);
     }
 printf("\n");
   }
}
}
```

OUTPUT :