JKKN COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE and Affiliated to Anna University) Natarajapuram, Komarapalayam, Namakkal-638 183.

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

Register Number……

Certified that this is the bonafide record of work done by Mr./Ms of ……..……..(year/semester)B.Tech He/She had done laboratory during the academic year 20 - 20 .

Signature of Lab In-charge **Signature of Head of the Department**

Submitted for the Anna University Practical Examination held on ---------- .

Internal Examiner **External Examiner**

TABLE OF CONTENTS

Sl.No	Bloom's level	Experiments Name	Relevance to COs	Page No
3C	Analyze	Program Using For Loop	CO1	
4A	Analyze	Program Using One Dimensional Array	CO1	
4B	Analyze	Program Using Two Dimensional Array	CO1	
4C	Analyze	Program Using Multi Dimensional Array	CO1	
4D	Analyze	Program Using Traversal	CO1	
5A	Analyze	Program Using String Operation to convert upper case to lower case	CO1	
5B	Analyze	Program Using String Operation to compare without string header file	CO1	
5C	Analyze	Program Using String Operation to compare with string header file	CO1	
6A	Analyze	Program Using Function call	CO1	
6B	Analyze	Program Using Return Statement	CO1	
6C	Analyze	Program Using Call by Value	CO1	

6D	Analyze	Program Using Call by Reference	CO1	
6E	Analyze	Program Using Passing Array to Function	CO1	
7A	Analyze	Program Using Direct Recursion	CO1	
7B	Analyze	Program Using Indirect Recursion	CO1	
8A	Analyze	Program Using Pointers with Function	CO1	
8B	Analyze	Program Using Pointers and Array	CO1	
8C	Analyze	Program Using Pointers and Strings	CO1	
8D	Analyze	Program Using Pointers and Pointers	CO1	
8E	Analyze	Program Using Array of Pointers	CO1	
9A	Analyze	Program Using Nested Structure	CO1	
9B	Analyze	Program Using Pointer to Structure	CO1	
9C	Analyze	Program Using Array of Structure	CO1	
9D	Analyze	Using Union	CO1	

10A	Analyze	Program Using Files-Reading and Writing from Files	CO1	
10B	Analyze	Program Using Files-File Pointers and File Operation	CO1	
10C	Analyze	Program Using Random Access File	CO1	
10D	Analyze	Program Using Processor Directives	CO1	

Ex. No : 1(A)	PROGRAM USING FORMATTED I/O STATEMENT
Date:	

AIM:

To Write C Program to print “Hello World!!!” using Formatting I/O Statement.

ALGORITHM:

Step-1: Start the Program

Step-2: Print the string “Hello World” using printf

statement. Step-3: Stop the Program

PROGRAM:

```
#include<stdio
.h>
#include<coni
o.h>  int  main
()
{
printf ("Hello
World!!!"); getch();
return 0;
}
```

ACCEPTED OUTPUT:

Hello World!!!

RESULT:

Thus the above C Program to print “Hello World!!!” using Formatting I/O Statement is executed successfully.

Ex. No : 1(B)	PROGRAMS USING UNFORMATTED I/O STATEMENT
Date:	

AIM:

To Write C Program to read (gets()) and display (puts()) the username using Unformatting I/O Statement.

ALGORITHM:

Step-1: Start the Program

Step-2: declared the character variable using array.

Step-3: Read the name of the user using gets() inbuilt Function. Step-4: Print the name of the user using puts() inbuilt Function Step-5: Stop the Program

PROGRAM:

```
#include<stdio
.h>
#include<coni
o.h> int main()
{
char
name[10];
gets(name)
;
puts(name)
; getch();
return 0;
}
```

SAMPLE INPUT:

KSR Institute for Engineering and Technology

EXCEPTED OUTPUT:

KSR Institute for Engineering and Technology

RESULT:

Thus the above C Program to read and display the username using Unformatting I/O Statement is executed successfully.

Ex. No : 1(C)	PROGRAMS USING EXPRESSIONS
Date:	

C Program to read values from the user and solve an expression. AIM:

To write a C program to read values from the user and solve an expression

ALGORTHIM:

Step- 1 : Start the Program

Step- 2 : Read the values of a and b.

Step- 3 : Compute the value of (a*a) + (2*a*b) +

(b*b). Step- 4 : Print the result.

Step- 5 : Stop the Program

PROGRAM:

```
#include<stdio
.h>
#include<coni
o.h>        void
main()
{
        int a,b,c;
        printf("Enter the values for a and
        b:"); scanf("%d%d",&a,&b);
        c=(a*a) + (2*a*b) + (b*b);
        printf("The result of the above expression is: %d",c);
}
```

SAMPLE INPUT:

Enter the values for a and b: 12 24

EXCEPTED OUTPUT:

The result of the above expression is: 1296

RESULT:

Thus the above C program to read values from the user and solve an expression is executed successfully.

Ex. No : 1(D)	PROGRAMS USING OPERATORS
Date:	

AIM:

To write a C Program to Perform Integer Division, Modulus Division, Normal Division and using Relational Operator, Equality Operators, Logical Operator.

ALGORTHIM:

Step- 1 : Start the Program

Step- 2 : Read the values of num1 and num2

Step- 3 : Compute the value of Integer Division, Modulus Division, Normal Division and also using Relational Operator, Equality Operators, Logical Operator

Step- 4 : Print the result. Step- 5 : Stop the Program

PROGRAM:

```
#include<stdio.h
   >
   #include<coni
   o.h>
void main()
    {
int
num1,num2,Integer_Division,Modulus_Division
; int Normal_Division,Less_Equal,And,Equal;
printf("ENTER THE NUMBER num1 and num2
:"); scanf("%d %d",&num1,&num2);
      Integer_Division = num1/num2;
      Modulus_Division= num1%num2;
      Normal_Division =
      (float)num1/num2;
printf("\n %d / %d =%d ",num1,num2,Integer_Division);
printf("\n %d % % %d =%d
",num1,num2,Modulus_Division); printf("\n %d / %d =%d
",num1,num2,Normal_Division); printf("
\n----------Relational Operator                    ");
    Less_Equal=num1<=num2;
```

```
printf("\n %d <= %d =%d ",num1,num2,Less_Equal);
```

```
    printf("\n----------Logical Operator                    ");
        And=(num1&&num2)+(num1||num2)+(!num1);
    printf("\nLogical Operator : % d",And);
    printf(" \n----------Equality Operator                    ");
        Equal=(num1==num2)||(num1!=num2);
    printf("\nEquality Operator :%d",Equal);
        }
```

SAMPLE INPUT:

ENTER THE NUMBER num1 and num2: 6 8

EXCEPTEDOUTPUT:

6/8=0

6%8=6

6/8=0

_____Relational Operator

6<=8=1

_____Logical Operator

_____ Logical Operator: 2

_____Equality Operator

_____ Equality Operator: 1

RESULT:

Thus the above C Program to Perform Integer Division, Modulus Division, Normal Division and also using Relational Operator, Equality Operators, and Logical Operator is executed successfully.

Ex. No : 2(A)	PROGRAMS USING IF AND IF ELSE STATEMENTS
Date:	

AIM:

To write C programs to implement decision-making constructs like IF, IF-ELSE.

ALGORITHM:

Step-1: Start the Program

Step-2: Read an integer from the user.

Step-3: Check whether the number is divisible

by 2. Step-4: If yes, print it is an even number.

Step-5: Else, print it is an odd

number. Step-6: Stop the Program

PROGRAM:

// using if-else statement

```
#include<stdi
o.h> void
main()
{
int num;
printf("Enter a
number:");
scanf("%d",&num);
if(num%2==0)
{
printf("\n It is an even number...");
}
else
{
printf("\n It is an odd number...");
}}
```

SAMPLE INPUT:

Enter a Number: 20

EXCEPTED OUTPUT:

It is an even number….

ALGORITHM:

Step-1: Start the Program

Step-2: Read the three values a,b and

c. Step-3: Check whether a is greater

than b.

Step-3.1:If yes, check whether a is greater than c. Step-3.1.1: If yes, print a is greater.

Step-3.1.2: Else, print c is greater.

Step-4: Else, check whether b is greater than c.

Step-4.1: If yes, print b is

greater. Step-4.2: Else, print c

is greater.

Step-5: Stop the Program

PROGRAM:

// using nested-if statements

```
#include<stdi
o.h> void
main()
{
int a,b,c;
printf("\n Enter three
integers:"); scanf("%d %d
%d",&a,&b,&c); if(a>b)
if(a>c)
printf("\n %d is
greater",a); else
printf("\n %d is
greater",c); else
if(b>c)
printf("\n %d is
greater",b); else
printf("\n %d is greater",c);
    }
```

SAMPLE INPUT:

Enter three integers: 23 89 67

EXCEPTED OUTPUT:

89 is greater

RESULT:

Thus the C programs to implement the various decision making constructs had been executed successfully.

Ex. No : 2(B)	PROGRAMS USING GOTO AND SWITCH STATEMENTS
Date:	

AIM:

To write a C programs to implement decision-making constructs using GOTO and SWITCH STATEMENT

ALGORITHM:

Step-1: Start the Program

Step-2: Label for goto statement as read.

Step-3: Enter the number between 1 and 999 from the user

Step-3.1:Read the number from the user and store it in num variable.

Step-3.1.1: Check, whether the entered number is not equal to 999, then go to step-3.1.2 , Otherwise go to step -4.

Step-3.1.2: if yes, check whether the entered number is less than 0, then goto read(Jump to label- read ,otherwise go to next step.

Step-3.1.3: calculate sum value.

Step-4: Else, print the sum of the number entered by user. Step-5: Stop the Program

PROGRAM:

```
#include<stdi
o.h> void
main()
{
intnum,sum
=0; read:
printf("Enter the number.Enter 999 to
end:"); scanf("%d", &num);
if(num!=999)
    {
```

```
if(num<
0) goto
read;
sum+=n
um;
goto
read;
        }
printf("\n Sum of the numbers entered by the user is = %d",sum);
}
```

SAMPLE INPUT:

Enter the number. Enter 999 to end:6 Enter the number.Enter 999 to end:89 Enter the number.Enter 999 to end:5 Enter the number.Enter 999 to end:67 Enter the number.Enter 999 to end:90 Enter the number. Enter 999 to end:999 **EXCEPTED OUTPUT:**

Sum of the numbers entered by the user is = 257

ALGORITHM:

Step-1: Start the Program.

Step-2: Read the grade from the user.

Step-3: Using switch case, suitable case statement is executed as per the grade given. Step-4: Print the mark range.

Step-5: Stop the Program.

PROGRAM:

// Grade System using Switch Case Statements

```
#include<stdi
o.h> void
main()
{
char grade;
printf("Enter the Grade you scored in C
programming: "); scanf("%c",&grade);
```

```
switch(grade)
        {
case 'S':
```

```
 printf("\n Your mark is between 91 and
 100"); break;
 case 'A':
 printf("\n Your mark is between 81 and
 90"); break;
 case 'B':
 printf("\n Your mark is between 71 and
 80"); break;
 case 'C':
 printf("\n Your mark is between 61 and
 70"); break;
 case 'D':
 printf("\n Your mark is between 56 and
 60"); break;
 case 'E':
 printf("\n Your mark is between 50 and
 55"); break;
 default:
 printf("\n OOPS, You have scored low marks in the exam... Study Well!!!");
        } }
```

SAMPLE INPUT:

Enter the Grade you scored in C programming: C

EXCEPTED OUTPUT:

Your mark is between 61 and 70

RESULT:

Thus the C programs to implement the various decision making constructs had been executed successfully.

Ex. No : 2(C)	PROGRAMS USING BREAK AND CONTINUE STATEMENT
Date:	

AIM:

To write a C programs to implement decision-making constructs using BREAKAND CONTINUE STATEMENT.

ALGORITHM:

Step-1: Start the Program

Step-2: Set the num variable, sum is equal to Zero.

Step-3: Check the while condition. if yes, go to next statement

Step-3.1:Enter the number between 1 and 999 from the user.

Step-3.2: Read the number from the user and store it in num variable.

Step-3.1.1: Check, whether the entered number is equal to 999, if yes, quit the loop and go to step4 Otherwise go to next step.

Step-3.1.2: Calculated the sum value.

Step-4: Else, print the sum of the number entered by user. Step-5: Stop the Program

PROGRAM:

```
#include<stdi
o.h> void
main()
{
intnum,su
m=0;
while(1)
  {
printf("Enter the number.Enter 999 to
stop:"); scanf("%d", &num);
if(num==999)
  {
break;
```

```
    }
sum+=num;
        }
```

printf("\n Sum of the numbers entered by the user is = %d",sum);

}

SAMPLE INPUT:

Enter the number.Enter 999 to stop:56 Enter the number.Enter 999 to stop:93 Enter the number.Enter 999 to stop:89 Enter the number.Enter 999 to stop:999

EXCEPTED OUTPUT:

Sum of the numbers entered by the user is = 238

ALGORITHM:

Step-1: Start the Program

Step-2: Initialize a variable num, sum=0, flag=0, count=0, avg.

Step-3: Check the while condition, flag is equal to 1. if yes, go to step 3.1otherwise,

go to step4Step-3.1:Enter any number between 1 and 999 from the user.

Step-3.2: Read the number from the user and store it in num variable.

Step-3.1.1: Check, if the num is equal to zero .if condition is true, skip the following statement. Otherwise go to step 3.1.2

Step-3.1.2: Check, if the num is not equal to 999. Then calculated sum value and count is increment by 1.if false, flag is assigned as zero.

Step-4: Else, print the sum of the number and

average of the number.Step-5: Stop the Program.

PROGRAM:

```
#include<stdio.h>
    void main()
    {
    intnum,sum=0,flag=1,coun
    t=0; float avg;
    while(flag==1)
       {
    printf("Enter the number.Enter 999 to stop:");
```

```
scanf("%d",
&num);
if(num==0)
continue;
if(num!=999)
        {
sum+=n
um;
count++
;
          }
els
e
fla
g=
0;
            }
printf("\n Sum =
%d",sum); avg=(float)
sum/count; printf("\n
Average = %f",avg);
}
```

SAMPLE INPUT:

Enter the number.Enter 999 to stop:34 Enter the number.Enter 999 to stop:89 Enter the number.Enter 999 to stop:67 Enter the number.Enter 999 to stop:56 Enter the number.Enter 999 to stop:0 Enter the number.Enter 999 to stop:56 Enter the number.Enter 999 to stop:3 Enter the number. Enter 999 to stop:999 **EXCEPTED OUTPUT:**

Sum = 305

Average = 50.833332

RESULT:

Thus the above C programs to implement decision-making constructs using Break and Continue has been Executed Successfully.

Ex. No : 3	PROGRAMS USING LOOPING STATEMENT
Date:	

AIM:

To write a C program to check whether the given number is an Armstrong number using while condition.

ALGORITHM:

Step-1: Start the Program.

Step-2: Read a value from the user in variable num.

Step-3: Initialize the value of sum as 0 and temp as

num. Step-4: Repeat until num is greater than 0.

Step-4.1: Compute digit=num%10 to extract the unit digit of

num. Step-4.2: Compute sum=sum+digit.

Step-4.3: Reduce num as num=num/10 to extract the digits except the last digit and goto

step-4. Step-5: Check whether the temp is same as sum.

Step-6: If yes, print it is an Armstrong

number. Step-7: Else, print it is not an

Armstrong number. Step-8: Stop the

Program.

PROGRAM:

// Armstrong

number

```
#include<stdio.h>
void main()
  {
intnum,temp,sum=0,d;
printf("\n Enter a
number:");
scanf("%d",&num);
temp=num;
while(num>0)
      {
        d=num%
10;
```

```
sum=sum+(d*d
*d);
```

```
num=num/10;
      }
if(sum==temp)
printf("\n It is an Armstrong
number..."); else
printf("\n It is not an Armstrong number..."); }
```

SAMPLE INPUT:

Enter a number: 371

EXCEPTED OUTPUT:

It is an Armstrong number...

B.C program to check whether the given year is a leap year using do while loop.

AIM:

To write a C program to check whether the given year is a leap year using do while loop

ALGORITHM:

Step-1: Start

Step-2: Read the value of a year.

Step-3: Check whether the year is divisible by 4 but not by 100. Step-4: Else check whether the year is divisible by 400.

Step-5: If either step-3 or step-4 is true, then print it is a leap year. Step-6: Else print it is not a leap year.

Step-7: Stop

PROGRAM:

```
#include<stdi
o.h> void
main()
{
int m=1900, n=1921;
do
   {
if(m%4
==0)
```

```
printf("\n %d is a leap
year",m); else
printf("\n %d is a not a leap
        year",m); m=m+1;
}while(m<=n);
   }
```

EXCEPTED OUTPUT:

```
1900 is a leap year
1901 is a not a leap
year 1902 is a not a
leap year 1903 is a
not a leap year 1904
is a leap year 1905
is a not a leap year
1906 is a not a leap
year 1907 is a not a
leap year 1908 is a
leap year 1909 is a
not a leap year 1910
is a not a leap year
1911 is a not a leap
year 1912 is a leap
year 1913 is a not a
leap year 1914 is a
not a leap year 1915
is a not a leap year
1916 is a leap year
1917 is a not a leap
year 1918 is a not a
leap year 1919 is a
not a leap year 1920
is a leap year 1921
is a not a leap year
```

C. C program to read the heights of the persons and find the average using for loop

AIM:

To write a C program to read the heights of the persons and find the average using for loop.

ALGORITHM:

Step-1: Start the Program

Step-2: Read the number of persons as „n‟. Step-3: Initialize the value of sum and i as 0. Step-4: Repeat until i is lesser than n.

Step-4.1: Read the height and store it in an array. Step-4.2: Add height to sum and store it in sum. Step-4.3: Add 1 to i and go to step-4.

Step-5: Compute the average of the array as sum/n.

Step-6: Print the persons with above the average height. Step-7: Stop the Program

PROGRAM:

```
#include<stdi
o.h> void
main()
  {
int
n,i,sum=0,count=0
; int height[10];
float avg;
printf("\n Enter the number of
persons:"); scanf("%d",&n);
printf("\n Enter the heights of
persons:\n"); for(i=0;i<n;i++)
        {
scanf("%d",&height[i])
; sum=sum+height[i];
```

```
        }
avg=sum/n;
```

```
for(i=0;i<n;i
++)
if(height[i]>
avg)
count=count
+1;
printf("\n The number of persons with above average height: %d",count);
}
```

SAMPLE INPUT:

```
Enter the number of
persons: 5 Enter the heights
of persons:
120
138
128
121
105
```

EXCEPTED OUTPUT:

The number of persons with above average height: 2

RESULT:

Thus the above C programs to implement Looping Statement have been Executed Successfully.

Ex. No : 4(A)	PROGRAMS USING ARRAY CONCEPTS- ONE DIMENSIONAL ARRAY
Date:	

AIM:

To write a program to implement a one dimensional array.

ALGORITHM:

Step 1: Start the

process. Step 2:

Initialize an array.

Step 3: Get each element of the array using a for

loop. Step 4: Display all the elements of the array.

Step 5: Stop the process.

PROGRAM:

```
#include<stdi
 o.h>
 voidmain ()
{
int a[5] ={15,30,45,60,75};
for (int i=0; i<5; i++){
printf("Value of one dimensional array a[%d]: %d \n",i,a[i]);
}
}
```

SAMPLE OUTPUT:

Value of arr [0]: 15

Value of arr [1]: 30

Value of arr [2]: 45

Value of arr [3]: 60

Value of arr [4]: 75

RESULT:

Thus the program to implement a one dimensional array is written and executed successfully.

Ex. No : 4(B)	PROGRAMS USING ARRAY CONCEPTS- TWO DIMENSIONAL ARRAY
Date:	

AIM:

To write a program to multiply two matrices using two dimensional array.

ALGORITHM:

Step- 1 : Start the program.

Step- 2 : Enter the value of m and n (or) order of the first matrix. **Step- 3 :**Enter the value of p and q (or) order of the second matrix. **Step- 4 :** Create a matrix of size a[m][n] and b[p][q].

Step- 5 : Enter the element of matrices row wise using loops.

Step- 6 : If a number of columns of the first matrix are not equal to the number of rows of the second matrix, print matrix multiplication is not possible and exit. If not, proceed to the next step.

Step- 7 : Create a third matrix, c of size m x q to store the product.

Step- 8 : Set a loop from i=0 to i=m.

Step- 9 : Set an inner loop for the above loop from j=0 to j=q.

Step- 10 : Initialise the value of element (i, j) of the new matrix to 0.

Step- 11 : Set an inner loop inside the above loop from k=0 to k=p.

Step- 12 : Using the add and assign operator (+=) store the value of a[i][k] * b[k][j] in the third matrix, c[i][j].

Step- 13 : Print the third matrix.

Step- 14 : Stop the program.

PROGRAM:

```
#include<stdi
o.h> void
main(){
int a[10][10],b[10][10],c[10][10],n,i,j,k;
printf("Enter the value of N(N<=10)
:"); scanf("%d",&n);
printf("Enter the elements of Matrix A:
```

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

```
for(j=0;j<n;j++){
scanf("%d",&a[i][j]
);
}
}
printf("Enter the elements of Matrix B:
\n"); for(i=0;i<n;i++){
for(j=0;j<n;j++){
scanf("%d",&b[i][j]
);
      }    }
for
(i=0;i<n;i++){
for
(j=0;j<n;j++){
c[i][j]=0;
for(k=0;k<n;k
++){
c[i][j]+=a[i][j]*b[k][j];
}
}
}
printf("Product of the two matrices is:
\n"); for (i=0;i<n;i++){
for(j=0;j<n;j++){
printf("%d\t",c[i][j]
);
}
printf("\n");
}
return ;0
}
```

SAMPLE INPUT:

```
Enter the value of
N(N<=10) :2 Enter the
```

elements of Matrix A: 1
2
3
4

Enter the elements of
Matrix B: 4
5
6
7

SAMPLE OUTPUT:

Product of the two
matrices is:

10 24

30 48

RESULT:

Thus the program to multiply two matrices using a two dimensional array was executed successfully.

Ex. No : 4(C)	PROGRAMS USING ARRAY CONCEPTS- MULTI-DIMENSIONAL ARRAY
Date:	

AIM:

To write a program to implement a multi-dimensional array.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Initialise an array.

Step- 3 : Get each element of the array using a for loop.

Step- 4 : Display all the elements of the array.

Step- 5 : Stop the Program.

PROGRAM:

```
#include<stdi
o.h>
voidmain()
{
    int arr[2][3][2]={{{1,2}{2,3},{3,4}},{{4,3},{3,2},{2,1}}};
for(int
i=0;i<2;i++){
for(int
j=0;j<2;j++){
for(int
k=0;k<2;k++){
printf("Values of arr[%d][%d][%d] is:%d \n",i,j,k,arr[i][j][k]);
                    } } } }
```

EXCEPTED OUTPUT:

```
Values of arr[0][0][0]
is:1 Values of
arr[0][0][1] is:2
Values of arr[0][1][0]
is:2 Values of
arr[0][1][1] is:3
Values of arr[1][0][0]
is:4 Values of
arr[1][0][1] is:3
```

Values of arr[1][1][0] is:3 Values of arr[1][1][1] is:2

RESULT:

Thus the program to implement a multi-dimensional array was executed successfully.

Ex. No : 4(D)	PROGRAMS USING ARRAY CONCEPTS- TRAVERSAL
Date:	

AIM:

To write a program to implement the traversal of arrays.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Initialize counter variable. Set i=Lower Bound (LB) **Step- 3 :** Repeat for i=Lower Bound (LB) to Upper Bound(UP) **Step- 4 :** Apply process to arr[i].

Step- 5 : Display the result

Step- 6 : Stop the Program.

PROGRAM:

```
#include<stdi
o.h> void
main()
{
int i, size;
int arr[]={1,-9,17,4,-3};
size=sizeof(arr)/sizeof(arr[0])
; printf("The array elements
are:"); for(i=0 ;i<size;i++){
printf("\narr[%d] =
%d",i,arr[i]);
}
}
```

EXCEPTED OUTPUT:

The array elements are:
arr[0] = 1
arr[1] = -9
arr[2] = 17
arr[3] = 4
arr[4] = -3

RESULT:

Thus the above program to implement the traversal of arrays was executed successfully.

Ex. No : 5(A)	PROGRAMS USING STRING OPERATION -CONVERT UPPER CASE TO LOWER CASE
Date:	

AIM:

To write programs to convert uppercase to lowercase.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Initialize the string space.

Step- 3 : Get the value using gets() inbuilt Function.

Step- 4 : Loop the values.

Step- 5 : Check the condition if(s[i]>=65&& s[i]<=90) and change the string into lower by s[i]+=32.

Step- 6 : Display the result.

Step- 7 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>
#include
<string.h>  void
main()
{
char
s[50];
int i;
printf("Enter the string
: "); gets(s);
for(i=0;s[i];i++)
{
if(s[i]>=65&&s[i]<
=90) s[i]+=32;
}
printf("string in lowercase ='%s'\n",s);
}
```

SAMPLE INPUT:

Enter the string: PROGRAMMING IN C PROGRAM

EXCEPTED OUTPUT:

String in lowercase ='programming in c program'

RESULT:

Thus the program to implement an uppercase to lowercase was executed successfully.

Ex. No : 5(B)	PROGRAMS USING STRING OPERATION -COMPARE TWO STRING WITHOUT USING STRING HEADER FILE
Date:	

AIM:

To Write a C Program in String comparison without using strcmp() function .

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declaration of two char array.

Step- 3 : Read the First and second string of the array from the user.

Step- 4 : Calling user defined Function named as compare().

Step- 5 : Comparing both the strings. **Step- 6 :** Integer variables declaration **Step- 7 :** While loop is executed.

Step- 8 : Go back to main function after while condition returned false.

Step- 9 : Compared the result using if statement

Step- 10 : Result is displayed.

Step- 11 : Stop the Program.

PROGRAM:

```
#include <stdio.h>
int
compare(char[],char[])
; void main()
{
  char
  str1[20];
  char
  str2[20];
  printf("Enter the first string
  : "); scanf("%s",str1);
  printf("Enter the second string
  : "); scanf("%s",str2);
  int c=
  compare(str1,str2);
  if(c==0)
  printf("Strings are
```

```
same"); else
printf("Strings are not same");
}
```

```
int compare(char a[],char b[])
 {
    int flag=0,i=0;
    while(a[i]!='\0'
    &&b[i]!='\0')
    {
      if(a[i]!=b[i])
      {
         fla
         g=
         1;
         bre
         ak;
      }
      i++;
    }
    if(flag=
    =0)
    return
    0; else
    return 1;
 }
```

SAMPLE INPUT:

Enter the first string : COMPUTER Enter the second string: COMputer

EXCEPTEDOUTPUT:

Strings are not same

RESULT:

Thus the program to implement a C Program in String comparison without using strcmp() function is executed successfully.

Ex. No : 5(C)	PROGRAMS USING STRING OPERATION -COMPARE TWO STRING USING STRING HEADER FILE
Date:	

AIM:

To Write a C Program in String comparison using strcmp() function .

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Initialise the values to compare. **Step- 3 :** Get two strings from the user. **Step- 4 :** Using library file strcmp().

Step- 5 : Display the result.

Step- 6 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>
#include<strin
g.h>          void
main()
{
int compare;
char str1[20]="COMPUTER SCIENCES";
char str2[20]="COMPUTER
sciences";
compare=strcmp(str1,str2);
if(compare==0)
printf("The strings are
equal"); else
if(compare>0)
printf("str1 is gr eater than
str2"); else
printf("str2 is gr eater than str1");
}
```

EXCEPTEDOUTPUT:

Str2 is greater than str1

RESULT:

Thus the program to implement to compare two strings using strcmp() function was executed

successfully.

Ex. No : 6(A)	PROGRAMS USING FUNCTION CONCEPT- FUNCTION CALL
Date:	

AIM:

Write the c program to find square using function call

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Include all required header files and declare all required variables.

Step- 3 : Write functions to perform square

Step- 4 : Get the value from the user to calculate the square and store it in “n”.

Step- 5 : Call the user-defined functions and display the result.

Step- 6 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>
intsquare(int
num)
{
return (num * num);
}
void main()
{
int
num
; int
n;
printf("Input any number for square
: "); scanf("%d", &num);
n = square(num);
printf("The square of %d is : %d\n", num, n);
}
```

SAMPLE INPUT:

Input any number for square: 4

EXCEPTED OUTPUT:

The square of 4 is: 16

RESULT:

Thus the program is for square using function is implemented and executed successfully.

Ex. No : 6(B)	PROGRAMS USING FUNCTION CONCEPT- RETURN STATEMENT
Date:	

AIM:

To writing a C Program to sum given values through returnstatements. .

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Include all required header files and declare all required variables.

Step- 3 : Write functions to perform sum of given values

Step- 4 : Sum the given numbers

Step- 5 : Call the user-defined

functions **Step- 6 :** Return the values

of the sum **Step- 7 :** Stop the

Program.

PROGRAM:

```
#include<stdi
o.h> int
sum();
void main()
{
   int addition;
   addition =
   sum();
   printf("\nSum of two given values = %d", addition);
}
 int sum()
{
   int a = 50, b = 80,
   sum; sum = a + b;
   return sum;
}
```

EXCEPTED OUTPUT:

Sum of two given values =130

RESULT:

Thus the program is for sum using function return is successfully executed.

Ex. No : 6(C)	PROGRAMS USING FUNCTION CONCEPT- CALL BY VALUE
Date:	

AIM:

To Write the C Program to perform swapping of two numbers using call by value.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Include all required header files and declare all required variables.

Step- 3 : Write functions to perform

Swapping **Step- 4 :** Call the values using

function call **Step- 5 :** Swap the given

values

Step- 6 : Stop the Program.

PROGRAM:

```
#include<stdio.
h> void
swap(int, int);
void main()
{
int a, b;
printf("Enter values for a and
b:\n"); scanf("%d%d", &a, &b);
printf("\n\nBefore swapping: a = %d and b = %d\n", a,
b); swap(a, b);
}
 void swap(int x, int y)
{
int
temp;
temp
= x;
        x   = y;
        y   = temp;
```

```
printf("\nAfter swapping: a = %d and b = %d\n", x, y);
}
```

SAMPLE INPUT:

Enter the values for a and b: 20 30

EXCEPTED OUTPUT:

Before swapping: a = 20 and b

= 30 After swapping: a= 30 and

b =20

RESULT:

Thus the above C Program to perform swapping of two numbers using call by value is executed successfully.

Ex. No : 6(D)	PROGRAMS USING FUNCTION CONCEPT- CALL BY REFERENCE
Date:	

AIM:

To Write the C Program to perform swapping of two numbers using call by Reference.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : : Include all required header files and declare all required variables.

Step- 3 : Write functions to perform

Swapping **Step- 4 :** Call the values using

function call **Step- 5 :** Swap the given

values

Step- 6 : Stop the Program.

PROGRAM:

```
#include<stdio.h
> void swap(int*,
int*); int main()
{
int a, b;
printf("Enter values for a and
b\n"); scanf("%d%d", &a, &b);
printf("\n\nBefore swapping: a = %d and b = %d\n", a,
b); swap(&a, &b);
printf("\nAfter swapping: a = %d and b = %d\n",
a, b); return 0;
}
void swap(int *x, int *y)
{
int temp;
temp =
*x;
*x = *y;
```

```
*y = temp;
```

```
}
```

SAMPLE INPUT:

Enter the values for a and b: 20 30

EXCEPTEDOUTPUT:

Before swapping: a = 20 and b
= 30 After swapping: a= 30 and
b =20

RESULT:

Thus the above C Program to perform swapping of two numbers using call by References is executed Successfully.

Ex. No : 6(E)	PROGRAMS USING FUNCTION CONCEPT- PASSING ARRAY TO FUNCTION
Date:	

AIM:

To Write a C program that passes arrays as function arguments.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare all required variables.

Step- 3 : Declare the function name with array as argument and array size

Step- 4 : Read the size of the array

Step- 5 : Call the function using array and size as arguments

Step- 6 : Use for loop to get the array elements

Step- 7 : Add the given

elements **Step- 8 :** Returns

value of addition **Step- 9 :**

Stop the Program.

PROGRAM:

```
#include<stdio.h>
int SumofNumbers(int a[], int Size)
{
int Addition
= 0; int i;
for(i = 0; i < Size; i++)
   {
      Addition = Addition + a[i];
   }
return Addition;
}
int main()
{
int i, Size,
a[10]; int
Addition;
```

```
printf("Please Enter the Size of an
Array: "); scanf("%d", &Size);
printf("\nPlease Enter Array
Elements\n"); for(i = 0; i < Size; i++)
   {
scanf("%d", &a[i]);
   }
   Addition = SumofNumbers(a,Size);
printf("Sum of All Elements in an Array = %d \n",
Addition); return 0;
}
```

SAMPLE INPUT:

Enter the size of an array: 5

Please enter array elements: 10 20 30 40 50

EXCEPTED OUTPUT:

Sum of all element in an array=150

RESULT:

Thus the above C Program that passes arrays as function arguments is executed successfully.

Ex. No : 7(A)	PROGRAMS USING RECURSION-DIRECT RECURSION
Date:	

AIM:

To write a program to demonstrate Fibonacci series using direct recursion.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare all required variables.

Step- 3 : Declare the function name with argument

Step- 4 : if the num i is equal to 0, return 0;

Step- 5 : If the num I is equal to 1 , return fibo_num (i - 1) + fibo_num (i -2);

Step- 6 : Use for loop to get the get first 10 fibonacci series

Step- 7 : Returns value of the fibonacci series.

Step- 8 : Stop the Program.

PROGRAM:

```
#include<stdio
.h> int
fibo_num (int
i)
{
if ( i == 0)
{
return 0;
}
if ( i == 1)
{
return 1;
}
return fibo_num (i-1)+fibo_num(i-2);
}
int main ()
{
int i;
for ( i = 0; i < 10; i++)
{
printf (" %d \t ", fibo_num (i));
}
return 0;
}
```

EXCEPTED OUTPUT:

0 1 1 2 3 5 8 13 21 24

RESULT:

Thus the above C Program for Fibonacci series using direct recursion is executed successfully.

Ex. No : 7(B)	PROGRAMS USING RECURSION-INDIRECT RECURSION
Date:	

AIM:

To demonstrate odd or even numbers using indirect recursion.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declaration of the odd and even() function

Step- 3 : Declare number as global variable

Step- 4 : IF statement check and execute the block till n is less than equal to 10

Step- 5 : Print the number by adding 1 for getting Even numbers

Step- 6 : Print the number by subtracting 1 for getting Odd

numbers **Step- 7 :** Call the odd function at once

Step- 8 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>  void
odd();
void
even();
int  num
=       1;
void
odd ()
{
if (num <= 10)
{
printf (" %d ", num +
1); num++;
even();
}
return;
}
void even ()
{
```

```
if ( num <= 10)
{
```

```
printf (" %d ", num
- 1); num++;
odd();
 }
return;
}
int main ()
{
odd();
return
0;
}
```

EXCEPTEDOUTPUT:

2 1 4 3 6 5 8 7 10 9

RESULT:

Thus the above C Program for odd or even numbers using indirect recursion is executed successfully.

Ex. No : 8(A)	PROGRAMS USING POINTERS-
Date:	POINTERS WITH FUNCTION

AIM:

To Write a C Program that pass pointer variable as parameters to functions.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the required variables and functions

Step- 3 : Read the values of a and b

Step- 4 : Assign the function named add to pointer variable ip

Step- 5 : Call the add function

Step- 6 : Sum the two variables and return the value

Step- 7 : Display the result

Step- 8 : Stop the Program.

PROGRAM:

```
#include
<stdio.h> int
add(int,int);
int main()
{
int a,b;
int
(*ip)(int,int)
; int result;
printf("Enter the values of a and b
: "); scanf("%d %d",&a,&b);
ip=add;
result=(*ip)(a,
b);
printf("Value after addition is :
%d",result); return 0;
}
```

int add(int a,int b)

```
{
int
c=a+b;
return
c; }
```

SAMPLE INPUT:

Enter the values of a and b: 5 10

EXCEPTEDOUTPUT:

Value after addition is: 15

RESULT:

Thus the above C Program that pass pointer variable as parameters to functions is executed successfully.

Ex. No : 8(B)	PROGRAMS USING POINTERS-
Date:	POINTERS AND ARRAY

AIM:

To Write a C Program to read and display an array of n integer with pointers concepts.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the array with size

Step- 3 : Initialize the array with

values **Step- 4 :** Declare the pointer

variable

Step- 5 : Assign the array of balance values to pointer variable

p **Step- 6 :** Iterate the loop and display the array values using

pointers **Step- 7 :** Iterate the array values using balance as

address

Step- 8 : Stop the Program.

PROGRAM:

```
#include
<stdio.h> int
main () {
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
double
*p; int
i;
  p = balance;
printf( "Array values using
pointer\n"); for ( i = 0; i < 5; i++ )
{
printf("*(p + %d) : %f\n", i, *(p + i) );
  }
printf( "Array values using balance as
address\n"); for ( i = 0; i < 5; i++ ) {
printf("*(balance + %d) : %f\n", i, *(balance + i) );
  }
return 0;
```

}

SAMPLE INPUT:

Array values using pointer

*(p + 0): 1000.000000

*(p + 1): 2.000000

*(p + 2): 3.400000

*(p + 3): 17.000000

*(p + 4): 50.000000

EXCEPTEDOUTPUT:

Array values using balance as address

*(balance + 0): 1000.000000

*(balance + 1): 2.000000

*(balance + 2): 3.400000

*(balance + 3): 17.000000

*(balance + 4): 50.000000

RESULT:

Thus the above C Program to read and display an array of n integer with pointers concept is executed successfully.

Ex. No : 8(C)	PROGRAMS USING POINTERS-
Date:	POINTERS AND STRING

AIM:

To write a C Program to read and print the string using Pointers

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the pointer variable with character array

Step- 3 : Initialize the array with values

Step- 4 : Iterate the loop and display the array values

Step- 5 : Stop the Program.

PROGRAM:

```
#include
<stdio.h> int
main()
{
char *cities[] = {"Salem",
"Chennai"}; int i;
for(i = 0; i < 2; i++)
printf("%s\n",
cities[i]); return 0;
}
```

EXCEPTED OUTPUT:

```
Sale

m

Che

nnai
```

RESULT:

Thus the above C Programto read and print the string using Pointers

Ex. No : 8(D)	PROGRAMS USING POINTERS- POINTERS TO POINTERS
Date:	

AIM:

To Write a C Program related with point to pointers and for declared the pointers to pointers, add (*) asterisk for each level of references.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the pointer variables

Step- 3 : Initialize the address of a to pointer variable p

Step- 4 : Display the output

Step- 5 : Stop the Program.

PROGRAM:

```
#include<stdi
o.h> void
main ()
{
int a =
10; int
*p;
int
   **p
   p;
   p =
   &a;
pp = &p;
printf("address of a: %x\n",p);
printf("address of p: %x\n",pp);
printf("value stored at p:
%d\n",*p); printf("value stored at
pp: %d\n",**pp);
}
```

EXCEPTEDOUTPUT:

address of a:

7cff1594 address

of p: 7cff1598

value stored at p:

10 value stored at

pp: 10

RESULT:

Thus the above C Program related with pointers to pointers is executed successfully.

Ex. No : 8(E)	PROGRAMS USING POINTERS-
Date:	ARRAYS OF POINTERS

AIM:

To write a C Program using an array of integer variable and to display the address of the identifiers.

ALGORITHM:

Step- 1 : Start the Program.
Step- 2 : Declare the pointer variable with an array
Step- 3 : Initialize the array index with pointer variable values
Step- 4 : Iterate the loop and display the address and array values the Program.
Step- 5 : Stop the Program.

PROGRAM:

```
#include<stdi
o.h> #define
SIZE 10 int
main()
{
int *arr[3];
int p = 40, q = 60, r =
90, i; arr[0] = &p;
arr[1] =
&q;
arr[2] =
&r;
for(i = 0; i < 3; i++)
{
printf("For the Address = %d\t the Value would be = %d\n", arr[i], *arr[i]);
}
return 0;
}
```

SAMPLE INPUT:

For the Address = 387130656 For the Address = 387130660 For the Address = 387130664

EXCEPTEDOUTPUT:

the Value would be = 40 the Value would be = 60 the Value would be = 90

RESULT:

Thus the above C Program using an array of integer variable and to display the address of the identifiers is executed successfully.

Ex. No : 9(A)	PROGRAMS USING STRUCTURE AND UNION- NESTED STRUCTURES
Date:	

AIM:

To write a C Program using structure to read and display employee name, employee id and date of joiningabout the employee.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the data members of structure

Step- 3 : Declare a structure variable and named as e1

Step- 4 : Assign the values of data members

Step- 5 : Display the output

Step- 6 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>
#include
<string.h>
struct
Employee
{
int id;
char
name[20];
struct Date
   {
intd
d;
int
mm
;
inty
yyy
;
}doj;
}e1;
int main( )
{
```

```
   e1.id = 437;
strcpy(e1.name,
"Praveen");
   e1.doj.dd = 29;
   e1.doj.mm = 03;
   e1.doj.yyyy = 2021;
printf( "employee id : %d\n", e1.id);
printf( "employee name : %s\n",
e1.name);
```

```
printf( "employee date of joining (dd/mm/yyyy) : %d/%d/%d\n",
e1.doj.dd,e1.doj.mm,e1.doj.yyyy); return 0;
}
```

EXCEPTEDOUTPUT:

```
employee id : 101
employee name :
Praveen
employee date of joining (dd/mm/yyyy) : 29/3/2021
```

RESULT:

Thus the above C Program using structure to read and display employee name, employee id and date of joining about the employee is executed successfully.

Ex. No : 9(B)	PROGRAMS USING STRUCTURE AND UNION- POINTER TO STRUCTURE
Date:	

AIM:

To Write a C Program using a pointer to a structure and to initialize the members of the structure.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the data members of structure **Step- 3 :** Display a structure variable with pointers **Step- 4 :** Assign the values of data members

Step- 5 : Display the output

Step- 6 : Stop the Program.

PROGRAM:

```
#include
<stdio.h>
#include
<string.h>
struct Subject
{
   // declare the member of the Course
structure charsub_name[30];
intsub_id;
charsub_duration[
50];
charsub_type[50];
};
int main()
{
struct   Subject
sub;        struct
Subject    *ptr;
ptr = &sub;
strcpy (sub.sub_name, " Computer
Science"); sub.sub_id = 1201;
strcpy (sub.sub_duration, "6 Months");
strcpy (sub.sub_type, " Multiple Choice
```

```
Question"); printf (" Subject Name: %s\t ",
(*ptr).sub_name); printf (" \n Subject Id: %d\t
", (*ptr).sub_id);
printf (" \n Duration of the Subject: %s\t ", (*ptr).sub_duration);
```

```
printf (" \n Type of the Subject: %s\t ",
(*ptr).sub_type); return 0;
}
```

EXCEPTED OUTPUT:

Subject Name: Computer

Science Subject Id: 1201

Duration of the Subject: 6 Months

Type of the Subject: Multiple Choice Questions

RESULT:

Thus the above C Program using a pointer to a structure and to initialize the members of the structure is executed successfully.

Ex. No : 9(C)	PROGRAMS USING STRUCTURE AND UNION- ARRAY OF STRUCTURE
Date:	

AIM:

To write a C Program to read and display the information of all the students in the class using array of structure.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the data members of structure

Step- 3 : Declare a structure variable and named as e1

Step- 4 : Assign the values of data

members **Step- 5 :** Read the array of

structure values **Step- 6 :** Iterate the

loop

Step- 7 : Display the output

Step- 8 : Stop the Program.

PROGRAM:

```
#include<stdio.
h> #include
<string.h>
struct student{
introllno;
char name[10];
};
int
main()
{ int i;
struct student st[5];
printf("Enter Records of 5
students"); for(i=0;i<5;i++){
printf("\nEnter
Rollno:");
scanf("%d",&st[i].roll
no); printf("\nEnter
Name:");
scanf("%s",&st[i].nam
```

```
e);
    }
printf("\nStudent Information
List:"); for(i=0;i<5;i++) {
printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);
```

```
    }
  return 0;
  }
```

SAMPLE INPUT:

```
  Enter Records of 5
  students Enter Rollno:1
  Enter
  Name:Sonoo
  Enter Rollno:2
  Enter
  Name:Ratan
  Enter Rollno:3
  Enter
  Name:Vimal
  Enter Rollno:4
  Enter
  Name:James
  Enter Rollno:5
  Enter
  Name:Sarfraz
```

EXCEPTEDOUTPUT:

```
  Student Information List:
  Rollno:1,
  Name:Sonoo
  Rollno:2,
  Name:Ratan
  Rollno:3,
  Name:Vimal
  Rollno:4,
  Name:James
  Rollno:5,
  Name:Sarfraz
```

RESULT:

Thus the above C Program to read and display the information of all the students in the class using array of structure is executed successfully.

Ex. No : 9(D)	PROGRAMS USING STRUCTURE AND UNION- UNION
Date:	

AIM:

To Write a C program to illustrate the concept of unions.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Define the union named number with two variables n1 and n2.

Step- 3 : Define the union variable x.

Step- 4 : Take the value of two variables using dot operator (i.e. x.n1, x.n2) as input.

Step- 5 : Print the values of two variables using dot operator as output.

Step- 6 : Stop the Program.

PROGRAM:

```
#include
<stdio.h> void
main()
{
union
number {
int n1;
float n2;
}; union
number x;
printf("Enter the value of n1:
"); scanf("%d", &x.n1);
printf("Value of n1 = %d",
x.n1); printf("\nEnter the
value of n2: "); scanf("%f",
&x.n2);
printf("Value of n2 = %f\n", x.n2); }
```

SAMPLE INPUT:

Enter the value of n1: 10 Enter the value of n2: 50

EXCEPTEDOUTPUT:

Value of n1 = 10

Value of n2 = 50.000000

RESULT:

Thus the above C Programto illustrate the concept of unions is executed successfully.

Ex. No : 10(A)	PROGRAMS USING FILES -
Date:	READING AND WRITING FROM THE FILE

AIM:

To write a C Program to read a file character by character, and display it simultaneously on the screen.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the file pointer variable

Step- 3 : Open a file

Step- 4 : Read the contents of a file

Step- 5 : Display the output

Step- 6 : Close the file

Step- 7 : Stop the Program.

PROGRAM:

```
#include<stdi
o.h> void
main( )
{
FILE
*fp ;
charch
;
char filename[20];
printf(“\n Enter the filename
:”); fp = fopen(filename,"r")
; if(fp==NULL)
{
printf(“\n Error Opening the
File”); exit(1);
}
ch=fgetc(fp)
;
while(ch!=E
OF)
{
```

```
putchar(c
h);
ch=fgetc(
fp);
}
Fclose(fp); }
```

SAMPLE INPUT:

Enter the filename: letter.TXT

EXCEPTEDOUTPUT:

Hello how are you?

RESULT:

Thus the above C Program to read a file character by character, and display it simultaneously on the screen is executed successfully.

Ex. No : 10(B)	PROGRAMS USING FILES -
Date:	FILE POINTERSAND FILE OPERATION

AIM:

To write a C Program to read some text from the keyboard and stored it in a file using pointer concept.

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the file pointer variable

Step- 3 : Open a file

Step- 4 : Read the contents of a file

Step- 5 : Write the new content into a file

Step- 6 : Display the output **Step- 7 :** Close the file **Step- 8 :** Stop the program.

PROGRAM:

```
#           include
<stdio.h>       #
include
<string.h>    int
main( )
{
      FILE *filePointer ;
      chardataToBeWritten[50] =
      "KIOT"; filePointer =
      fopen("college.c", "w") ; if (
      filePointer == NULL )
      {
                                                printf( "college.c file failed to open." ) ;

      }
      else
      {
                                                p
                                                r
                                                i
```

```
n
t
f
(
"
T
h
e
f
i
l
e
i
s
n
o
w
o
p
e
n
e
d
.
\
n
"
)
;
i
f
(
s
t
r
l
e
n
(
d
a
t
a
T
o
B
e
W
r
i
t
t
e
n
)
>
0
)
{
fputs(dataToBeWritten,
filePointer) ; fputs("\n",
filePointer) ;
```

```
        }
        fclose(filePointer) ;
        printf("Data successfully written in file
        college.c\n"); printf("The file is now closed.") ;
    }
    return 0;
}
```

EXCEPTEDOUTPUT:

The file is now
opened.

File Writing

Data successfully written in file

college.c The file is now closed.

RESULT:

Thus the above c program C Program to read some text from the keyboard and stored it in a file using pointer concept is executed successfully.

Ex. No : 10(C)	PROGRAMS USING FILES -
Date:	FILE RANDOM ACCESS

AIM:

To write a C Program to read about function that are used to randomly access a record stored in a binary file using fseek(),rewind(),ftell().

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the file pointer variable

Step- 3 : Open a file

Step- 4 : Read the contents of a file

Step- 5 : Write the new content into a file

Step- 6 : Display the output **Step- 7 :** Close the file **Step- 8 :** Stop the Program.

PROGRAM:

Fseek()

Program

```
#include
<stdio.h> void
main(){
  FILE *fp;
fp =
fopen("myfile.txt","w+");
fputs("This is college",
fp); fseek(fp, 7,
SEEK_SET ); fputs("cse
department", fp);
fclose(fp);
}
```

EXCEPTEDOUTPUT:

This is cse

department

rewind()

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Declare the file pointer variable

Step- 3 : Open a file

Step- 4 : Read the contents of a file

Step- 5 : using rewind function moves the file pointer at beginning of the file

Step- 6 : Display the

output **Step- 7 :** Close

the file **Step- 8 :** Stop

the Program.

PROGRAM:

```
#include<stdi
o.h> void
main(){ FILE
*fp;
char c;
fp=fopen("file.txt",
"r");
while((c=fgetc(fp))!=EOF){
printf("%c",c);
}
rewind(fp);//moves the file pointer at beginning of the
file while((c=fgetc(fp))!=EOF){
printf("%c",c);
}
fclose(fp);
}
```

EXCEPTEDOUTPUT:

this is a simple textthis is a simple text

ftell()

ALGORIT

HM:

Step- 1 : Start the Program.

Step- 2 : Declare the file pointer variable

Step- 3 : Open a file

Step- 4 : Read the contents of a file

Step- 5 : Find the length of the file using ftell function

Step- 6 : Close the file

Step- 7 : Display the

output **Step- 8 :** Stop the

program

PROGRAM

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

```
void
  main
  (){
  FILE
  *fp;
int length;
fp = fopen("file.txt",
"r"); fseek(fp, 0,
SEEK_END); length
= ftell(fp); fclose(fp);
printf("Size of file: %d bytes", length);
}
```

Sample Output :

```
Size of file: 21 bytes
```

RESULT:

Thus the above C Program to read about function that are used to randomly access a record stored in a binary file using fseek(),rewind(),ftell() is executed successfully.

Ex. No : 10(D)	PROGRAMS -
Date:	PROCESSOR DIRECTIVES

AIM:

To write a C Program using preprocessor directives

ALGORITHM:

Step- 1 : Start the Program.

Step- 2 : Define the value of macro as 3.14

Step- 3 : Display the output

Step- 4 : Stop the Program.

PROGRAM:

#define

Program

```
#include
<stdio.h>
#define PI
3.14 void
main() {
printf("%f",PI)
;
}
```

Output

3.140000

Algorithm related with

macro Step 1: Start the program

Step 2: Define the value of macro as 0

Step 3: Display the output as 0 if macro is assigned as 0

Step 4: Stop the program

#if Program

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

```
NUMBER 0
void main() {
#if (NUMBER==0)
printf("Value of Number is:
%d",NUMBER); #endif
getch();
```

}

Output

Value of Number is: 0

#else

Algorit

hm

Step 1: Start the program

Step 2: Define the value of macro as 1

Step 3: Display the output as 0 if macro is assigned as 0, Otherwise display the output as non-zero

Step 4: Stop the program

Program

```
#include
<stdio.h>
#define
NUMBER 1
void main() {
#if NUMBER==0
printf("Value of Number is:
%d",NUMBER); #else
print("Value of Number is
non-zero"); #endif
}
```

Output

Value of Number is non-zero

RESULT:

Thus the above C Program using preprocessor directives is executed successfully and verified.