Experiment No:-

Experiment Name:- Write a C program to perform all basic operations on string.

Objective:- The objective of this program is to perform all basic operations on string.

Algorithm:-

Step-1:- start

Step-2:- set str1[12] = "Hello "
str2[12] = "World"
str3[12]

step-3:- strcpy(str3, str1)

step-4:- Print str3

Step-5:- strcat(str1, str2)

Step-6:- Print str1

step-7:- len = strlen(str1)

Step-8:- Print len

Step-9:- cmp = strcmp(str1, str2)

step-10:- If cmp == 0 then goto step-11
else goto Step-12

step-11:- Print "strings are Equal" and goto step-13

Step-12:- Print "strings are Unequal" and goto step-13

Step-13:- Print cmp

step-14:- End

Source Code:-

```
#include<stdio.h>
#include <string.h>
int main()
{
char str1[12] = "Hello ", str2[12] = "World", str3[12];
int len, cmp;
strcpy(str3, str1);
printf("strcpy(str3, str1): %s\n", str3);
strcat(str1, str2);
printf("strcat(str1, str2): %s\n", str1);
len = strlen(str1);
printf("strlen(str1): %d\n", len);
cmp = strcmp(str1, str2);
if(cmp == 0)
    printf("Strings are Equal\n");
else printf("Strings are Unequal\n");
printf("Value Returned by strcmp() is: %d\n", cmp);
return 0;
}
```

Sample Input/Output:-

```
strcpy(str3, str1): Hello
strcat(str1, str2): Hello World
strlen(str1): 11
Strings are Unequal
Value Returned by strcmp() is: -1
```

Discussion:- By this program we can perform all basic operations on string.

Experiment No: 01

Experiment Name: Write a program to input and display an array.

Objectives: This program is create to input and display an array.

Algorithm:

Step 1: Start

Step 2: Int a[5], i, S=0;

Step 3: Let i=0

Step 4: Calculate i= i+1 if i<5 and go to step 5

Step 5: Calculate S= S+a[i]

Step 6: for i<5 calculate i++ and print i, a[i]

step 7: print S

step 8: Stop

Source Code:

```
#include <stdio.h>
main()
{ int a[5],i, s=0;
  printf("Enter the numbers: \n");
 for(i=0; i<5; i++)
 { scanf("%d", &a[i])
     s=s+a[i];
  }
 printf("\n Numbers are in: \n");
for(i=0; i<5; i++)
 { printf("a[%d]=%d\n", i, a[i]);
}
 printf("\n S=%d", s);
}
```

Sample Input:

Enter the number

1
2
3
4
5

Sample Output:

Number are in

a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5

Discussion: By this program, we can understand the scanf and printf.

Experimet No: 02

Experiment Name: Write a program to insert an element into an array.

Objectives : This program is to create to ~~insert~~ insert an element into an array.

Algorithm:

INSERT (LA, N, K, ITEM)

1. [Initialize counter] Set J := N
2. Repeat steps 3 and 4 while J ≥ K
3. [Move Jth element downward] Set LA[J+1] := LA[J]
4. [Decrease Counter] Set J := J-1

   [End of step 2 loop]
5. [INSERT elements] set LA[K] := ITEM
6. [Reset N] Set N := N+1
7. Exit.

Source Code:

```
#include <stdio.h>
int main ()
{ int array [100], position, c, n, value;
printf("Enter number of values of element \n");
Scanf ("%d", &n);
printf ("Enter %d elements \n");
for(c=0; c<n; c++)
  Scanf("%d", &array[c]);
printf("Enter the location where you wish to insert an element\n");
  Scanf("%d", &position);
 printf("Enter the value to insert \n");
     Scanf ("%d", &value);
   for ( c=n-1; c>=position-1; c--)
     array [c+1] = array [c];
   array [position-1] = value;
   printf ("Resultant array is \n");
   for(c=0; c<=n; c++)
   printf ("%d", array [c]);
   return 0;
}
```

Sample Output:

```
Enter number of elements in array
5
Enter 5 elements
2
5
4
3
8
Enter the location where wish to insert an element
4
Enter the value to insert
10
Resultant array is
2
5
4
10
3
8
```

Discussion: By this program, we know about scanf and printf

Experiment No: 03

Experiment Name: Write a program to delete an element from an array.

Objectives: This program is create to delete an element from an array.

Algorithm:

DELETE(LA, N, K, ITEM)

1. Set ITEM := LA [K]
2. Repeat for J = K to N-1
   [Move J+1st element upward] Set LA [J] := LA [J+1]
   [End of loop]
3. [Reset the number N of element in LA] Set N := N-1
4. Exit.

Source Code:

```
#include <stdio.h>
Void main ( )
{ int a[100], i, n, pos;
  printf("\n Enter no of elements \n");
  Scanf ("%d", &n);
  printf ("Enter the elements \n");
  for(i=0; i<n; i++);
  {
     Scanf("%d", & a[i]);
  }
  printf("Element of array are \n");
  for(i=0; i<n; i++)
  { printf ("a[%d] = %d\n", i, a [i]);
  }
  printf(" Enter the position from which the number has to be
          deleted \n");
  Scanf("%d", & pos);
  for(i= pos; i<n-1; i++)
  { a[i] = a[i+1];
  } n=n-1;
    printf ("\n On deletion, new array we get is \n");
    for (i=0; i<n; i++)
    { printf("a[%d] = %d\n", i, a [i]);
    }
}
```

Sample Output:

Enter no of elements
5
Enter the elements
7 8 12 3 9
Elements of array are
a[0] = 7
a[1] = 8
a[2] = 12
a[3] = 3
a[4] = 9
Enter the position from which the number has to be deleted
3
On deletion, new array we get is
a[0] = 7
a[1] = 8
a[2] = 12
a[3] = 9

Discussion: By this, we can understand the printf and scanf.

Experiment No : 04

Experiment Name: Write a program to search an element in an array using Linear search algorithm.

Objectives : This program is create to search an element in an array using Linear search algorithm.

Algorithm : This algorithm finds the location LOC of ITEM in DATA, or sets LOC := 0, if the search is successful.

LINEAR (DATA, N, ITEM, LOC)

1. [Insert ITEM at the End of DATA]
   Set DATA [N+1] = ITEM
2. [Initialize counter] Set LOC := 1
3. [Search for ITEM]
   Repeat while DATA [LOC] = /ITEM;
   Set LOC := LOC + 1
   [END of Loop]
4. [Successful] if LOC = N+1, then : Set LOC := 0.
5. Exit.

Source Code:

```
#include <stdio.h>
#include <conio.h>
 void main()
{
 int main item, la[19]={0,12,84,64,7,47,11}, LOC=1, n=6;
 printf("Enter the item = ");
 scanf("%d", &item);
 la[n+1] = item;

while (la[loc] != item)
{ loc = loc+1;
 }
 if (LOC == (n+1))
   { loc = 0;
     printf ("LOC = %d", loc);
 }
else
  printf ("\n\n LOC = %d", loc);
 printf ("\n Your given item = %d", item);

}
 getch();
 }
```

Sample Input : Enter the item = 13

Sample Output : LOC = 13

Your given item = 13

Discussion : By this program, we can search an element from an array using linear search algorithm.

Experiment No: 05

Experiment Name: Write a program to search an element in an array using binary search algorithm.

Objectives : This program is to create to search an item from an array using binary search algorithm.

Algorithm :

BINARY (DATA, LB, UB, ITEM, LOC)

1. [initialize segment variables]
   Set BEG := LB, END := UB and MID = INT((BEG + END) / 2)
2. while (BEG <= END and DATA [MID] = / ITEM)
3. If (ITEM < DATA [MID]) then :
   Set END = MID - 1
   ELSE :
   Set BEG := MID + 1
   [End of if structure]
4. Set MID := INT ((BEG + END) / 2)
   [END of step 2 loop]
5. If (DATA [MID] = ITEM) then :
   Set LOC := MID

Else

Set LOC := NULL

[End of if structure]

6. Exit.

Source Code:

```
#include<stdio.h>
#include<conio.h>
 void main()
{  clrscr();
int item, mid, beg, end, data[20]={0,91,92,93,94}, lb=1, ub=7;
 beg=lb, end=ub, mid=int((beg+end)/2);
printf("Enter the item= ");
Scanf("%d", &item);
while(data[mid]!=item && beg<=end)
{if(item<data[mid])
   end=mid-1;
else
     beg=mid+1;
  mid=int((beg+end)/2);
}
 if(data[mid]==item)
{ printf("\n LOC = %d", mid);
printf("\n\n Your given item is= %d", item);
}
```

```
else
 { mid = 0;
   printf ("LOC = %d", mid);
 }
 getch();
}
```

Sample input : Enter the item = 91

Sample output : LOC = 1; Your given item is = 91.

Discussion: By this program, we can search an element from an array by using binary search algorithm.

Experiment No: 06

Experiment Name: Write a program to sort an array using bubble short algorithm.

Objectives: This program is to create to sort an array using bubble sort algorithm.

Algorithm:

BUBBLE (DATA, N)

1. Repeat step 2 and 3 for K=1 to N-1.
2. Set PTR := 1. [initialization]
3. Repeat while [PTR <= N-K]:
   a) if DATA [PTR] > DATA [PTR+1], then;
      interchange DATA [PTR]
   [End of if structure]
   b) Set PTR := PTR+1.
   [End inner loop]
   [End step 1 outer loop]
4. Exit.

Source Code:

```
#include<stdio.h>
# include<conio.h>
void main()
{
int k,ptr, data[33]={0,32,51,27,85,66,23,13}, n=8, temp;
for (k=1; k<n; k++)
{ for (ptr=0; ptr<n-k; ptr++)
 { if (data[ptr] > data[ptr+1])
   { temp = data[ptr];
     data[ptr]= data[ptr+1];
     data[ptr+1]= temp;
   }
     }
}
printf ("\n\n The sorted data is= ");
for (k=0; k<n; k++)
printf("%d", data[k]);
getch();
}
```

Sample Input : 0, 32, 51, 27, 85, 66, 23, 13

Sample output :

The sorted data is = 0, 13 23 27 32 51 66 85

Discussion: By this program, we can sort an array using bubble short algorithm.

Experiment No: 07

Experiment Name: Write a program to sort an array using quick short algorithm.

Objective: This program is create to sort an array using quick sort algorithm.

Algorithm:

1. [Initialize] TOP := NULL
2. [Push boundary values of A onto stacks when A has 2 or more elements]
   If N > 1, then TOP := TOP+1, LOWER[1] := 1, UPPER[1] := N.
3. Repeat step 4 to 7 while TOP ≠ NULL
4. [Pop sublist from stacks]
   Set BEG := LOWER[TOP], END := UPPER[TOP]
   TOP := TOP-1
5. Call QUICK(A, N, BEG, END, LOC).
6. [Push left sublist onto stack when it has 2 or more elements]
   If BEG < LOC-1, then:
   TOP := TOP+1, LOWER[TOP] := BEG
   UPPER[TOP] = LOC-1
   [End of if structure]
7. [Push right sublist onto stacks when it has 2 or more elements]
   If LOC+1 < END, then;
   TOP := TOP+1, LOWER[TOP] := LOC+1

UPPER [TOP] := END

[End of if structure]

[End of if structure]

8. Exit.

Source Code:

```
#include<stdio.h>
# Include<iostream.h>
#Include <conio.h>
#Include< stdlib.h>

void quick(int a[], int left, int right)
{ int i=left, j=right, mid;
  mid = (a[i] + a[j])/2;
  while (i<=j)
     { while (a[i]<mid)
          i++;
       while (a[j]>mid)
          j--;
    if (i<=j){
          int temp;
          temp = a[i];
          a[i] = a[j];
          a[j] = temp;
          i++;
          j--;
   }

}  if (left<j)
       quick (a, left, j);
```

```
if (j< right)
        quick (a,i,right);
}
void main()
   int n,a[100];
   cout<<"How many Elements you want to enter: ");
   cin >>n;
   for (int i=0; i<n; i++)
      {  cin>>a[i];
   }
   quick (a,0,n-1);
   cout<<"\n after Quick Sort : ";
   for (i=0; i<n; i++)
    {
       cout<<"\n\t"<<a[i];

   }
   getch();
}
```

Sample Input: Enter the number of array

9 2 3

Sample output: after Quick Sort

2 3 9

Discussion: By this program, we know about cin and cout.

Experiment No: 07

Experiment Name: Write a program to sort an array using quick sort algorithm.

Objectives: This program is create to sort an array using quick sort algorithm.

Algorithm:

Step-1: Consider the first element of the list as pivot.

Step-2: Define two variables i and j. Set i and j to first and last element of the list respectively.

Step-3: Increament i until list[i] > pivot then stop.

Step-4: Decreament j until list[j] < pivot then stop.

Step-5: If i < j then exchange list[i] and list[j].

Step-6: Repeat step 3, 4 & 5 until i > j

Step-7: Exchange the pivot element with list[j] element.

Source Code:

```
#include<stdio.h>
void quick_sort(int [],int,int);
int partition (int [] ,int,int);
int main ()
{
   int a[50],i,j;
   printf("How many elements ? ");
   scanf("%d",&n);
   printf("\n Enter array element : ");
   for(i=0 ;i<n;i++)
   scanf("%d",&a[i]);
   quick_sort (a,0,n-1);
   printf ("\n Array after sorting : ");
   for (i=0 ;i<n;i++)
   printf ("%d", a[i]);
   return 0;
  }
void quick_sort (int a[],int l,int u]
{ int j;
  if (l<u).
{ j= partition (a,l,u);
   quick_sort (a,l,j-1);
   quick_sort (a,j+1,u);
 }}
int partition (int a[], int l,int u)
{ int v,i,j, temp;
   v=a[l];
   i=l;
   j=u+1;
 do
 { do
```

```
    i++;
  while (a[i]<v && i<=u);
  do
  j--;
  while (v<a[j]);
  if(i<j)
{ temp=a[i];
   a[i]=a[j];
  a[j]=temp;
}while(i<j);
  a[l]=a[j];
  a[j]=v;
 return(j);
}
```

Sample Output:

How many element? 6

Enter array element: 5 6 2 3 1 7

Array after sorting: 1 2 3 5 6 7

Discussion: By this program, we can understand the printf and scanf.

Experiment No: 08

Experiment Name: Write a program to find sum of two matrices.

Objectives: This program is to create to find the sum of two matrices.

Algorithm:

Step 1: Start

Step 2: Declare matrix A [r] [c];
and matrix B [r] [c];
and matrix C [r] [c]
r = no. of rows, c = no. of colours

Step 3: Read r, c, A[], B[][]

Step 4: Declare variable i = 0, j = 0

Step 5: Repeat until i<r
5.1: Repeat until j<c
C[i][j] = A[i][j] + B[i][j]
Set j = j+1
5.2: Set i = j+1

Step 6: C is the require matrix after addition.

Step 7: Stop

Source Code:

```
#include<stdio.h>
int main()
{  int  r, c, i, j, A[10][10], B[10][10], sum[10][10];
printf("Enter the number of rows and colouns of matrix");
scanf ("%d%d", &r, &c);
printf(" Enter the element of A matrix\n");
  for (i=0; i<r ; i++)
    for (j=0 ; j<c ; j++)
        scanf ("%d", &A[i][j]);
printf("Enter the element of second matrix\n");
     for(i=0 ; i<r ; i++)
       for (j=0;j<c;j++)
         scanf("%d", &B[i][j]);
printf ("Sum of entered matrices :- \n");
      for (i=0; i<r; i++)
        { for(j=0; j<c ; j++)
            { sum[i][j] = A[i][j]+B[i][j];
   printf("%d\t", sum[i][j]);
   }  printf ("\n");
  }  return 0;
}
```

Sample Input: Enter the number of rows and colum of matrix 2 2

Enter the element of A matrix

1 2
3 4

Enter the element of B matrix

5 6
7 1

Sample Output: Sum of entered matrices:-

6 8
10 5

Discussion: By this program, we can add two matrices.

Experiment No: 09

Experiment Name: Write a program to multiply two matrices.

Objectives: This program is create to multiply two matrices.

Algorithm:

Step 1 : Start

Step 2 : Declare matrix A[m][n]

and matrix B[p][q]

and matrix C[m][q]

Step 3 : Read m, n, p, q

Step 4 : Now check the matrix can be multiplied or not, if n is not equal to q matrix can't be multiplied and an error message is generated.

Step 5 : Read A[][] and B[][]

Step 6 : Declare variable i=0, k=0, j=0 and sum=0

Step 7 : Repeat step until i<m

7.1 : Reapt step until j<q

7.1.1 : Repeat until k<p

Set sum = sum + A[i][k] * B[k][j]

Set multiply[i][j] = sum;

Set sum = 0 and k=k+1

7.1.2 : Set j=j+1

7.2 : Set i = i+1

Step 8 : Q is the required matrix

Step 9 : Stop.

Source Code:

```
#include <stdio.h>
int main()
{ int m,n, p,q, i, j, k, sum=0;
int A[10][10], B[10],[10], C[10][10];
printf("Enter the number of row and coloum of
        A matrix \n");
scanf("%d%d", &m, &n);
printf(" Enter the element of the A matrix \n");
 for (i=0; i<m; i++)
for (j=0; j<n; j++)
scanf("%d", &A[i][j]);
printf(" Enter the number of row and coloum of
        B matrix \n");
scanf("%d %d", &p, &q);
if (n!=p)
printf(" Matrices with entered orders can not be
        multiplied with each other \n");
else
{
   Printf ("Enter the element of B matrix \n");
   for (i=0; i <p; i++)
   for (j=0; j<q; j++)
```

```
    scanf("%d",&B[i][j]);
for (i=0; i<m; i++)
{ for (j=0; j<q; j++)
  { for (k=0; k<p; k++)
{ sum = sum + A[i][k] * B[k][j];
 }
 multiply [i][j] = sum;
 sum = 0;
}
}
printf("Product of the entered matrices: \n");
for(i= 0; i <m; i++)
{ for (j= 0; j<q; j++)
 printf("%d \t", multiply [i][j]);
  print f ("\n");
}
 }
  }
 return 0;
}
```

Sample Input: Enter the number of rows and coloums of A matrix 3 3

Enter the elements of A matrix

1 2 0
0 1 1
2 0 1

Enter the number of rows and coloum of B matrix 3 3

Enter the elements of B matrix

1 1 2
2 1 1
1 2 1

Sample Output : Product of entered matrices:

5 3 4
3 3 2
3 4 5

Disoussion: By this program, we can multiply two matrices.

Experiment No: 10

Experiment Name: Write a program to implement all operation in a linked list (traversing, insertion and deletion).

Objectives: This program is create to implement all operation in a linked list (traversing, insertion and deletion)

Algorithm:

(Traversing a linked list)

1. Set PTR := START [Initialize pointer PTR]
2. Repeat steps 3 and 4 while PTR ≠ NULL
3. Apply process to INFO [PTR]
4. Set PTR := LINK [PTR]

   [End of step 2 loop]
5. Exit.

(Inserting)

INSLOC (INFO, LINK, START, AVAIL, LOC, ITEM)

1. [OVERFLOW?] If AVAIL = NULL, then write: Overflow and exit.
2. [Remove first node from AVAIL list]

   Set NEW := AVAIL and AVAIL := LINK [AVAIL]
3. Set INFO [NEW] := ITEM
4. If LOC = NULL, then [INSERT as first node]

Set LINK[NEW] := START and START := NEW

Else [Insert after a node with Location LOC]

set LINK[NEW] := LINK[LOC] and [LINK][LOC] := NEW

[End of if structure]

5. Exit.

(Delete)

DEL (INFO, LINK, START, AVAIL, LOC, LOCP)

1. If LOCP = NULL, then:

Set START := LINK[START]. [Delete first node].

else:

Set LINK[LOCP] := LINK[LOC] [Delete node N]

[End of if structure]

2. Set LINK[LOC] := AVAIL and AVAL := LOC

3. Exit.

Source Code:

```
#include <stdlib.h>
#include <iostream.h>
using namespace std;
Struct node
{ int data;
  Struct Node * next;
};
void insertAtBeginning (struct Node** head_ref, int
 {                        new_data)
   Struct Node* new_node = (Struct Node*) malloc (sizeof
                                   (struct node));
   new_node -> data = new_data;
   new_node ->next = (*head_ref);
   (*head_ref) = new_node;
}
Void insertAfter (Struct Node* prev_node, int new_data).
  {if (prev_node == NULL)
   { cout << " The previous node can't be NULL";
     return;
  }
   Struct Node* new_node = (Struct * Node*) malloc;
   new_node -> data = new_data;
   new_node -> next = prev_node -> next;
   prev_node -> next = new_node;
}
```

```
void InsertAtEnd (struct Node **head_ref, int new_data)
{
  struct Node *new_data = (struct Node *) malloc (sizeof
                                          (struct Node));
  struct Node *last = *head_ref;
  new_node -> data = new_data;
  new_node -> next = null;
  if (*head_ref == NULL)
  { *head_ref = new_node;
    return;
  }
  while (last -> next != NULL) last = last -> next;
  last -> next = new_node;
  return;
}
Void delete node (struct Node ** head_ref, in key)
{
   Struct Node *temp = *head_ref, *prev;
   if (temp != NULL && temp -> data = key)
   { *head ref = temp -> next;
    free (temp);
    return;
   }
    while (temp != NULL & temp -> data != key)
   { prev = temp;
    temp = temp -> next;
   }
   if (temp == NULL) return;
```

```
    prev->next = temp->next;
    free(temp);
}

Struct node* temp = head;
cout << "\n list element are: ";
  while (temp != NULL)
    {
      cout << "%d ..... ";
       cin >> temp_data;
     temp = temp-> next;
    }

Int main()
{
   Struct node* head = NULL;
    insert At End (&head, 1);
   Insert At Beginning (& head, 2);
   insert At Bejinning (&head, 3);
   Insert At End (&head, 4);
   Insert After (head->next, 5);
   cout << " linked list : ";
   print list (head);

  cout << "\n After deleting an element : ";
   deletenode(&head, 3);
   print list (head);
}
```

Sample Output:

Enter a number to insert

10

Enter a number to insert

10

Enter data to be insert

30

Inserted item = 10 10 30

Delete data to eleminent

30

New data = 10 10

Discussion: By this program, we can understand the cout and cin

Experiment No: 11

Experiment Name: Write a program to implement all operation at stack.

Objectives: This program is create to implement all operation at stack.

Algorithm:

Algorith (Push):

PUSH (STACK, TOP, MAXSTK, ITEM)

1. [STACK ALREADY filled ?]
   If TOP:=MAXSTK, then Print: Overflow, and return.
2. Set TOP: =TOP+1. [Increasing Top by 1]
3. Set STACK [TOP] := ITEM
4. Return.

Algorithm (Pop):

Pop (STACK, TOP, ITEM)

1. If TOP = 0, then, Print: Underflow and return.
2. Set ITEM: = STACK (TOP]
3. Set TOP: = TOP-1 [Decrease Top by 1]
4. Return.

Source Code:

```
#include<limits.h>
#include <stdio.h>
#include <stdlib.h>
struct stack{
    int top;
    unsigned capacity;
    int *array;
};
struct stack *createstack(unsigned capacity)
{ struct stack *stack (struct stack*)malloc (sizeof(struct
                                                Stack));
  stack->capacity = capacity;
  stack->top = -1;
  stack->array = (int*) malloc(stack -> capacity * sizeof (int));
  return stack;
}
int isfull( struct stack *stack)
{ return stack > Top == -1;
}
 void push (struct stack * stack, int item)
 { if (isfull (stack))
       return 0;
   stack->array [++ stack-> Top] =item;
   printf ("%d", pushed to stack \n", item).;
}
 int pop (struct stack * stack)
   {
```

```
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top--];
}
int peek(struct stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top];
}
int main()
{
    struct stack* stack = createStack(100);
    push(stack, 90);
    push(stack, 50);
    push(stack, 40);
    printf("%d popped from stack\n", pop(stack));
    return 0;
}
```

Sample Output:

```
90 pushed into stack
50 pushed into stack
40 pushed into stack
40 poped from stack
Top element is: 50
Elements present in stack : 50 90.
```

Discussion: By this program, we can understand about scanf and printf.

Experiment No: 12

Experiment Name: Write a program to implement all operation of queue.

Objectives: This program is create to implement all operation of queue.

Algorithm:

QINSERT (QUEUE, N, FRONT, REAR, ITEM)

1 [Queue is already filled]
If FRONT = 1 and REAR = N, or if FRONT = REAR+1 ; then:
Write: Overflow and return.

2. [Find new value of REAR]
If FRONT := NULL , then: [Queue initially empty]
Set FRONT := 1 and REAR := 1
Else if REAR = N , then:
Set REAR := 1
Else:
Set REAR := REAR + 1.
[End of if structure]

3. Set QUEUE [REAR] := ITEM [This insert new element]

4. Return.

QDELETE (QUEUE, N, FRONT, REAR, ITEM)

1. [Queue already empty ?]
   If FRONT := NULL, then : WRITE : Underflow and return.
2. Set ITEM := QUEUE [FRONT]
3. [Find new value of FRONT]
   If FRONT = REAR, then : [Queue has only one element to start.]
      Set FRONT := NULL and
      REAR := NULL
   Else if FRONT := N then.
      Set FRONT := 1
   Else :
      Set FRONT := FRONT+1.
   [End of if structure and]
4. Return.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
void insert ();
void delete ();
int queue_array [MAX];
int rear = -1;
int front = -1;
{ int choice;
  while (1)
{ printf ("1. Insert element to queue \n");
  printf ("2. Delete element from queue \n");
  printf ("Enter your choice: ");
  scanf ("%d", &choice);
 switch (choice)
 { case 1:
    insert() break;
   case 2:
    delete()
    break; default:
  printf ("Wrong choic \n");
}
}
void insert()
 { int item;
  if (rear == MAX-1)
```

```
    printf("Queue Overflow \n");
  else
  { if (front==-1)
    front=0;
    printf("Insert the element in queue : ");
    scanf("%d", &item);
    rear = rear+1;
    queue_array[rear] = item;
  }

  void delete()
  { if front==-1 || front > rear)
    {
      printf("Queue Underflow \n");
      return;
    }else
    { printf("Element delete from queue is : %d\n", queuearray
                                                    [front]);
    front = front+1;
    }
```

Sample Output:

1. Insert element to queue
2. Delete element from queue

Enter your choice: 1

Insert the element into queue: 4

Queue is: 4

Enter your choice: 2

Delete the element from stack: 4

Queue is: Underflow

Discussion: By this program, we can understand the scanf and printf.

Experiment No: 13

Experiment Name: Write a program to implement binary tree and its traversing algorithm (pre-order, in-order, post-order)

Objectives: This program is create to implement binary tree and its traversing algorithm (pre-order, in-order, post-order).

Algorithm:

Traversing Algorithm:

PREORD (INFO, LEFT, RIGHT, ROOT)

1. [Initially push NULL onto stack and initialize PTR]
   Set TOP := 1, STACK[1] := NULL and PTR := ROOT
2. Repeat steps 3 to 5 while PTR ≠ NULL
3. Apply process to INFO[PTR]
4. [RIGHT child?]
   If RIGHT[PTR] ≠ NULL, then: [PUSH on stack]
      Set TOP := TOP+1, and STACK[TOP] := RIGHT[PTR]
5. If LEFT[PTR] ≠ NULL, then:
      Set PTR := LEFT[PTR].
   Else: [Pop from STACK]
      Set PTR := STACK[TOP] and TOP := TOP-1
6. Exit.

INORD (INFO, LEFT, RIGHT, ROOT)

1. [Push NULL onto stack and initialize PTR]
   Set TOP := 1, STACK[1] := NULL and PTR := ROOT
2. Repeat while PTR ≠ NULL.
   (a) Set TOP := TOP+1
   (b) Set PTR := LEFT[PTR]
3. Set PTR := STACK[TOP] and TOP := TOP-1.
4. Repeat step 5 to 7 while PTR ≠ NULL
5. Apply PROCESS to INFO[PTR]
6. If RIGHT[PTR] ≠ NULL, then:
   (a) Set PTR := RIGHT[PTR]
   (b) Go to step 3.
7. Set PTR := STACK[TOP] and TOP := TOP-1.
8. Exit.

POSTORDER (INFO, LEFT, RIGHT, ROOT)

1. Set TOP := 1, STACK[1] := NULL and
   PTR := ROOT
2. Repeat steps 3 to 5 while PTR ≠ NULL.
3. Set TOP := TOP+1 and STACK[TOP] := PTR
4. If RIGHT[PTR] ≠ NULL, then [push on stack]
   Set TOP := TOP+1 and
   STACK[TOP] := -RIGHT[PTR]

5. Set PTR := LEFT [PTR].

6. Set PTR := STACK [TOP] and TOP := TOP-1

7. Repeat while PTR > 0

   (a) Apply PROCESS to INFO [PTR]

   (b) Set PTR := STACK [TOP] and TOP := TOP - 1

8. If PTR < 0, then

   (a) Set PTR := -PTR

   (b) Go to step 9

9. Exit.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  char data;
  struct node *right_child;
  struct node *left_child;
};
struct node *new_node(char data)
{
  struct node *p;
  p= malloc (sizeof(struct node));
  p->data =data;
  p->left_child = NULL;
  p->right_child = NULL;
  return(p);
}
void preorder(struct node *root)
{
  if(root!=NULL)
  {
    printf("%c ",root->data);
    preorder(root->left_child);
    preorder(root->right_child);
  }
}
void postorder(struct node *root)
{
  if(root!=NULL)
  {
    postorder(root->left_child);
    postorder(root->right_child);
    printf("%c",root->data);
```

```
}
}
void in-order (struct node * root)
{if (root != NULL)
{
   inorder (root -> left child);
   printf ("%c", root -> data);
  inorder (root -> right child);
   }
}
int main()
{
   Struct-node * root;
   root = new_node ('D');
   root -> left_child = new_node ('A');
   root -> right_child = new_node ('F');
   root -> left-child -> left child = new_node ('E');
   root -> left_child -> right_child = new_node ('B');
   preorder (root);
   printf ("\n");
  postorder (root);
  printf ("\n");
  inorder (root);
  printf ("\n");
  return 0;
}
```

Sample Output:

D A E B F
E B A F D
E A B D F

Discussion: By this program, we can understand the printf and scanf?

Experiment No: 14

Experiment Name: Write a program to implement tower of Hanoi.

Objectives: This program is create to implement tower of Hanoi.

Algorithm:

TOWER (N, BEG, AUX, END)

This program gives a recursive solution to the Tower of Hanoi problem for N disks.

1. If N = 1, then;
   (a) Write: BEG → END
   (b) Return.
   [End of If structure]
2. [Move N-1 disks from peg BEG to peg AUX]
   Call TOWER (N-1, BEG, END, AUX)
3. Write: BEG → END
4. [MOVE N-1 disks from peg AUX to peg END]
   Call TOWER (N-1, AUX, BEG, END)
5. Return.

Source Code:

```
#include <stdio.h>
void towers (int, char, char, char);
int main ()
{ int num;
  printf ("Enter number of disks : ");
  scanf ("%d",&num);
  printf ("The operation on sequence of move involved in
          the towers of Hanoi are :\n");
   towers (num, 'A', 'C', 'B');
   return 0;
}
void towers (int num, char frompeg, char topeg, char
             auxpeg)
{
   if (num == 1)
{  printf ("\n Move disk 1 from peg %c to peg %c", frompeg,
          topeg);
    return 0;
}
  towers (num-1, frompeg, auxpeg, topeg);
 printf ("\n Move disk %d from peg %c to peg %c",
                    num, frompeg, topeg);
```

towers (num-1, auxpeg, topeg, frompeg);

}

Sample Output:

Enter the number of disk: 3

The operation or sequence of move involved in the Tower of Hanoi are:

Move disk 1 from peg A to peg C.

Move disk 2 from peg A to peg B

Move disk 1 from peg C to peg B

Move disk 3 from peg A to peg C

Move disk 1 from peg B to peg A

Move disk 2 from peg B to peg C

Move disk 1 from peg A to peg C.

Discussion: By this program, we can understand the scanf and printf.

Experiment No: 15

Experiment Name: Write a program to implement first pattern matching algorithm.

Objectives: This program is create to implement first pattern matching algorithm.

Algorithm:

P and T are strings with R and S length respectively and are stored as arrays with one character per element.

1. [Initialize] Set k:=1 and MAX:= S-R+1
2. Repeat steps 3 to 5 while k ≤ MAX.
3. Repeat for L= 1 to R [Tests each character of P]
   If P[L] ≠ T[k+L-1], then go to step 5.
   [End of inner loop]
4. [Success] Set INDEX=K, and Exit.
5. Set k:=k+1
   [End of step 2 outer loop]
6. [Failure] Set INDEX=0
7. Exit.

Source Code:

```
#include<stdio.h>
#include<string.h>
int cmatch(char[], char[]);
int main(){
  char a[100], b[100];
  int position;
  printf("Enter Some text\n");
  gets(a);
  printf("Enter a string to find\n");
   gets(b);
  position = match(a,b);
if (position != -1)
{ printf("Found at location: %d\n", position+1);
 }
else{
    printf("Not found.\n");
}
  return 0;
}
int match(char, text[], char pattern[])
{ int c,d,e, text_length, pattern_length, position=-1;
   text_length = Strlen(text);
   pattern_length= strlen(pattern);
if(pattern_length > text_length)
{
   return -1;
```

```
}
  for (c=0 ; c<= text_length - pattern_length, c++)
{ position = e = c;
   for(d=0; d< pattern_length ; d++)
   { if (pattern [d] == text [e])
      { e++;
      }
       else {
     }    break;
   }
  if (d== pattern_length) {
      return position ;
  }
  }
   return -1;
}
```

Sample Output: Enter some text
Computer Programming is fun.
Enter a string to find
programming is fun.
Found at location 10.

Discussion: By this program, we can understand the printf and scanf.

Experiment No: 16

Experiment Name: Write a program to implement all string operation.

Objectives: This program is create to implement all string operation.

Algorithm:

Source Code:

```
#include<stdio.h>
#include<string.h>
int main(){
char str1[12] = "Harry";
char str2[12] = " Potter";
char str3[12];
int len;

strcpy(str3, str1);
printf("strcpy(str3, str1): %s\n", str3);

strcat(str1, str2);
printf("strcat(str1, str2): %s\n", str1);

len = strlen(str1);
printf("strlen(str1): %d\n", len);

int res = strcmp(str1, str2);
if(res == 0)
    printf("Strings are equal");
else
    printf("Strings are unequal");

printf("\n Value returned by strcmp() is: %d", res);

return 0;
}
```

Sample Output:

Strcpy(str3, str1) : Happy

Strcpy (str1, str2) : HappyPattern

Strlen(str1): 11

Strings are unequal

Value returned by strcmp() is : -1

Discussion: By this program, we can understand printf and scanf.

Experiment No: 17

Experiment Name: Write a program to replace a substring into a text.

Objectives: This program is create to replace a substring into a text.

Algorithm:

A text T and pattern P and Q are in memory. This algorithm replace every occurance of P in T by Q.

1. [Find index of P] Set K := INDEX (T, P)
2. Repeat while K ≠ 0:
   (a) [Replace P by Q] Set T := REPLACE (T, P, Q)
   (b) [update index] Set K := INDEX (T, P)
   [End of loop]
3. Write : T
4. Exit.

Source Code:

```
#include <iostream.h>
#include <cstring>
using namespace std;
int main()
{
  string str, str2, str3;
  cout << "Enter the main string: ";
  cin >> str;
  cout << "Enter the string to be replaced:";
  cin >> str2;
 int str2len = str2.length();
 cout << "Enter the replacing string: ";
 cin >> str3;
 str.replace(str.find(str2), str2.size(), str3);
 cout << str;
 return 0;
}
```

Sample Output:

Enter main string : Hogwartismyhome

Enter the string to be raplace : home

Enter the replacing string : life

Output -

Hogwartismylife.

Discussion: By this program, we can understand the cout and cin.

Experiment No: 18

Experiment Name: Write a program to determine n Fibbonacci numbers using recursive.

Objectives: This program is create to determine n Fibbonacci number using recursive.

Algorithm:

FIBONACCI (FIB, N)

1. If N=0 or N=1, then (set FIB := N) and return.
2. Call FIBONACCI (FIBA, N-2)
3. Call FIBONACCI (FIBB, N-1)
4. Set FIB := FIBA + FIBB
5. Return.

Source Code:

```
#include <bits/stdc++.h>
using namespace std;
int fib (int n)
{
  if (n <= 1)
     return n;
  return fib(n-1) + fib(n-2);
}
int main ()
{
  int n = 9;
  cout << fib(n);
  getchar();
  return 0;
}
```

Sample Output: 34

Discussion: By this program, we can understand the cin and cout.

Experiment No: 19

Experiment Name: Write a program to calculate factorial of a number using recursive.

Objectives: This program is to create to calculate factorial of a number using recursive.

Algorithm:

Step-1 : Start

Step-2 : Declare variable n, fact, i

Step-3 : Read the number from user

Step-4 : Initialize variable fact = 1 and i = 1

Step-5: Repeat Until i<=number

5.1 fact = fact * i

5.2 i = i + 1

Step6: Print fact

Step-7: Stop

Source Code:

```
#include<iostream>
using namespace std;
int fact (int n){
  if((n==0)||(n==1))
    return 1;
  else
    return n * fact (n-1);
}int main (){
      int n=4;
      cout<<"factorial of "<<n<<" is "<<fact(n);
      return 0;
}
```

Sample Output: 24

Discussion: By this program, we can understand cin and cout.