DATA STRUCTURE MANUAL

1. Write a program to find GCD using recursion ?

```
#include<stdio.h>
#include<conio.h>
int gcd(int a,int b);
void main()
{
        int a,b, ans;
        clrscr();
        printf("\n enter two numbers:");
        scanf("%d %d",&a,&b);

        ans=gcd(a,b);

        printf("\n GCD=%d", ans);
        getch();
}
int gcd(int a,int b)
{
        if(a==b) return(a);
        else
        if(a>b)
          return(gcd(a-b,b));
        else
          return(gcd(a,b-a));
}
```

Algorithm

Step 1 − Define the recursive function.

Step 2 − Read the two integers a and b.

Step 3 − Call recursive function.

a. if i>j

b. then return the function with parameters i,j

c. if i==0

d. then return j

e. else return the function with parameters i,j%i.

Output:

```
enter two numbers:10 25
GCD=5
```

2. Write a program to find Binomial Coefficient Using Recursion

```
#include<stdio.h>
#include<conio.h>
int fact(int n);
void main()
{
        int n,r,ans;
        clrscr();

        printf("\n enter n and r values:");
        scanf("%d %d",&n ,&r);

        ans=fact(n)/(fact(r)*fact(n-r));

        printf("\n binomial coefficient %d C %d= %d",n ,r, ans);
        getch();
}
int fact(int n)
{
        if(n==0)
            return (1);
        else
            return(n*fact(n-1));
}
```

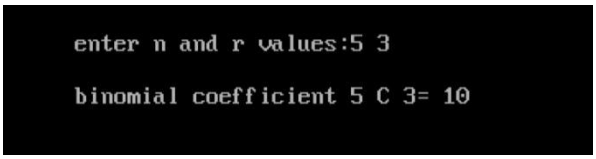
Algorithm Pascal triangle

1. start
2. input n (number of rows)
3. repeat : for i=0;i<n;i++
   repeat : for c=0;c<=(n-i-2) ;c++
   print " "
   repeat :c=0;c<=i;c++
   print (fact(i)/(fact(c)*f))

Algorithm function fact(n)

1. input c
2 result=1
3 repeat : for c=1;c<=n;c++
   result=result*c
4 return (result)

Output:



3. Write a program illustrate tower of Hanoi program for N-disk?

```
#include<stdio.h>
#include<conio.h>
void tower(int n, char A,char B,char C);
int count =0;
void main()
{
        int n;
        clrscr();

        printf("\n enter number of Disks :");
        scanf("%d", &n);

        tower(n, 'A' ,'B','C');

        printf("\n number of moves=%d", count);
        getch();
}
void tower( int n, char A, char B, char C)
{
        if(n==1)
{
        printf("move disk from %c -> %c\n ", A , C);
        count++;
}
        else
{
```

```
        tower(n-1,A,C,B);
        tower(1,A,B,C);
        tower(n-1,B,A,C);
 }
 }
```

Algorithm tower of hanoi()

1. START
2. Input n (number of disks)
3. call function tower(n, s,i,d)

4 stop

Algorithm function tower(n, src, inter, dest)

1 IF n == 1, THEN
  move disk from source to dest
 ELSE
 2. tower(n - 1, src, dest,inter)
 3. move disk from source to dest
 4. tower(n - 1, inter,src, dest)
 END IF
 END function

Output:

```
enter number of Disks :3
move disk from A -> C
move disk from A -> B
move disk from C -> B
move disk from A -> C
move disk from B -> A
move disk from B -> C
move disk from A -> C
number of moves=7
```

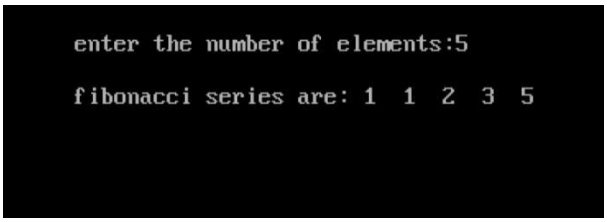
4. Write a program to find Fibonacci numbers Using Recursion?

```
#include<stdio.h>
#include<conio.h>
int fib(int n);
void main()
{
        int i,n;
        clrscr();
        printf("\n enter the number of elements:");
        scanf("%d",&n);
        printf("\n fibonacci series are: ");
        for(i=1;i<=n;i++)
        {
                printf("%d ",fib(i));
        }
        getch();
}
int fib(int n)
{
        if(n==0)
          return(0);
        else
         if(n==1)
            return(1);
        else
          return(fib(n-1)+fib(n-2));
```

Algorithm of Fibonacci Series

Step 1: START
Step 2: Declare variable n1, n2, sum, n, i
Step 2: Initialize variables:
n1 = 0, n2 = 1, i = 2
Step 3: Read n
Step 4: Repeat this step until i <= n:
sum = n1 + n2
print sum
n1 = n2
n2 = sum
i = i + 1
Step 5: STOP

Output:



5. Write a program to find largest and smallest element in the given array elements.

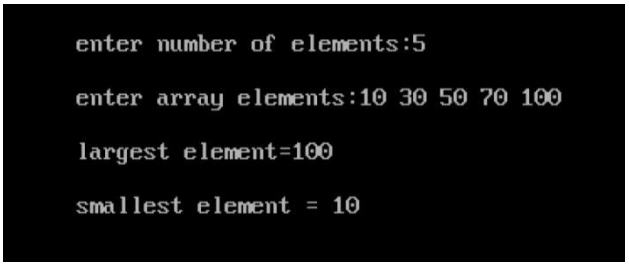
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10],n ,i,small ,large;
        clrscr();

        printf("\n enter number of elements:");
        scanf("%d", &n);

        printf("\n enter array elements:");
        for(i=0;i<n ; i++)
        scanf("%d",&a[i]);

        large=a[0];
        small=a[0];

        for(i=1;i<n; i++)
        {
            if (a[ i ]>large)
                large =a[ i ];
        else
         {
                if(a[i]<small)
                small=a[i];
```

```
        }
 }
        printf("\n largest element=%d\n ", large);
        printf("\n smallest element = %d",small);
        getch();
 }
```

Algorithm:

Step1: start
Step2:Declaring variables n, I, min, max, *ptr
Step3: read value of n
Step4: Allocate memory
Ptr←(int *)malloc(n* sizeof(int))
Step5: read elements from i←0 to n
Read ptr[i]
Step6: Intialize min,max to ptr[0]
Min←ptr[0]
Max←ptr[0]
Step7: do i←0 to n
If(ptr[i]>max)
Max←ptr[i]
If(ptr[i]<min)
min←ptr[i]
Step8: Display Max element is max
Display Min Element is min.
Step9: stop

Output:



6. Write a program to search the given key element using linear search .

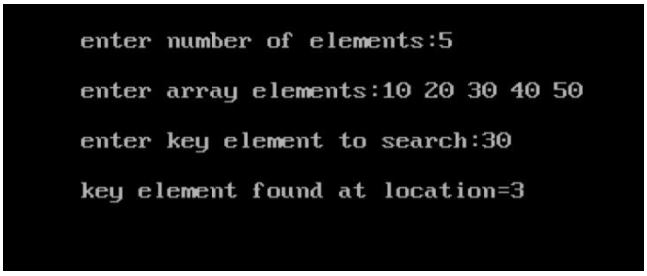
```
#include<stdio.h>
#include<conio.h>
int lsearch(int a[],int n,int key);
void main()
{
        int a[10],n,i,key,loc;
        clrscr();

        printf("\n enter number of elements:");
        scanf("%d",&n);

        printf("\n enter array elements:");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);

        printf("\n enter key element to search:");
        scanf("%d",&key);

        loc=lsearch(a,n,key);

        if(loc==-1)
                printf("\n key element not found");
```

```
        else
                printf("\n key element found at location=%d", loc);
                getch();
}
int lsearch(int a[],int n,int key)
{
        int i;
        for(i=0;i<n;i++)
        {
                if(key==a[i])
                return(i+1);
        }
        return(-1);
}
```

ALGORITHM

STEP 1:[START]
Program to search Linear elements
STEP 2:[INPUT]
n,a[1],search,
Step3:[condition/loop]
for(i=0;i<n;i++)
if(a[i]==search)
STEP4:[OUTPUT]
Search
STEP5: [STOP]
END

Output:



7. Write a program to search a given key element using binary search.

```
#include<stdio.h>
#include<conio.h>
int bsearch(int a[],int n,int key);
void main()
{
        int a[10],n,i,key,loc;
        clrscr();

        printf("\n enter number of elements:");
        scanf("%d",&n);

        printf("\n enter array elements:");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);

        printf("\n enter key element to search:");
        scanf("%d",&key);

        loc=bsearch(a,n,key);

        if(loc==-1)
                printf("\n key element not found");
        else
                printf("\n key element found at location=%d", loc);
        getch();
}
```

```
int bsearch (int a[], int n ,int key)
{
        int  low, high, mid;
        low =0;
        high =n-1;
        while (low<=high)
        {
                mid=(low+high)/2;
                        if(key ==a[mid])
                                return(mid+1);
                        else
                                if(key<a[mid])
                                        high=mid-1;
                                else
                                        low=mid+1;
        }
        return(-1);
}
```

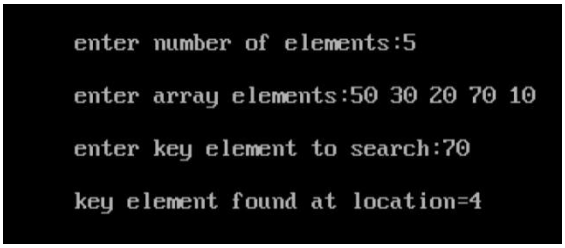
Algorithm:

Step 1 : Start
Step 2 : Declare Array and read Array Elements in Ascending order
Step 3 : Display Array Elements
Step 4 :Read Key Element
Step 5 : Call function Binary_search(a,0,size,key)
Step 6 : Stop
Step 1 : start
Step 2 : called Binary _search(A,lo,hi,x)
Step 3 :If(lo>hi) then
Return ;
Step 4 :Mid<-(lo+hi)/2

Step 5 :If(x==a[mid]) then
Return mid

Else if
(X<-a[mid]) Then
Return binary_search(a,lo,mid-1,key)
Else
Return binary_search(a,mid+1,hi,key)
Step 6:stop

Output:



8. Write a program to sort given elements using bubble sort

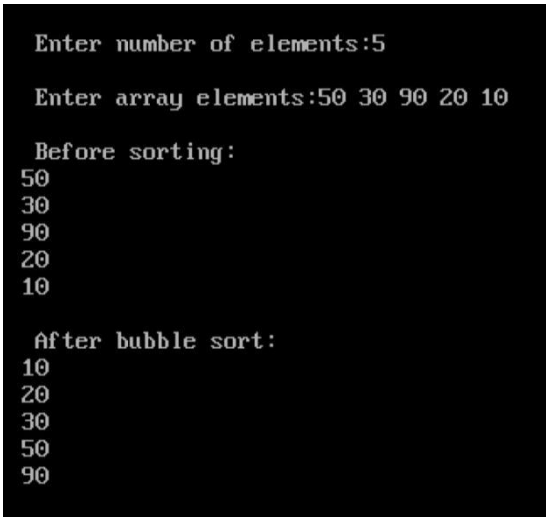
```
#include<stdio.h>
#include<conio.h>
void bubblesort(int a[],int n);
void main()
{
        int a[10],n,i;
        clrscr();

        printf("\n enter number of elements:");
        scanf("%d",&n);

        printf("\n enter array elements:");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);

        printf("\n before sorting:");
        for(i=0;i<n;i++)
        printf("\n%d",a[i]);

        bubblesort(a,n);

        printf("\n after bubble sort:");
        for(i=0;i<n;i++)
        printf("\n%d",a[i]);
        getch();
}
```

```
void bubblesort(int a[],int n)
{
        int i,j,temp;
        for(i=1;i<n;i++)
        {
                for(j=0;j<n-i;j++)
                {
                        if(a[j]>a[j+1])
                                {
                                        temp =a[j];
                                        a[j]=a[j+1];
                                        a[j+1]=temp;
                                }
                }
        }
}
```

Algorithm:

1.start
2.input array n,c,d, swap
3. repeat for c=0;c<n:c++
   read array elements
4. repeat c=0;c<n-1;c++
   repeat d=0;d<n-c-1;d++
   repeat if array[d]>array[d+1]
   swap=array[d]
5.repeat c=0;c<n;c++
  print array elements
6.return 0
7.stop

Output:


Enter number of elements:5
Enter array elements:50 30 90 20 10
Before sorting:
50
30
90
20
10
After bubble sort:
10
20
30
50
90

9. Write a program to sort the given n- elements using insertion sort.

```
#include<stdio.h>
#include<conio.h>
void insertionsort(int a[],int n);
void main()
 {
        int a[10],i,n;
        clrscr();
        printf("\n enter number of elements:\n");
        scanf("\n%d",&n);
        printf("\n enter array elements:\n");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        printf("\n before sorting:");
        for(i=0;i<n;i++)
        printf("\n %d",a[i]);
        insertionsort(a,n);
        printf("\n after sorting insertion :");
        for(i=0;i<n;i++)
        printf("\n %d",a[i]);
        getch();
}
void insertionsort(int a[],int n)
{
    int i,j,temp;
```

```
    for(i=1;i<n;i++)
     {
        for(j=i;j>0;j--)
         {
            if(a[j]<a[j-1])
              {
                  temp=a[j];
                  a[j]=a[j-1];
                  a[j-1]=temp;
              }
           }
       }
}
```

Algorithm:

Step 1: Start

Step 2:declare bubblesort function

Step 3: input a[],n,i,temp;

Step 4:for(i=1;i<n;i++)

{

Step 5:for(j=0;j<n;j++)

{

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

{

temp=a[j];

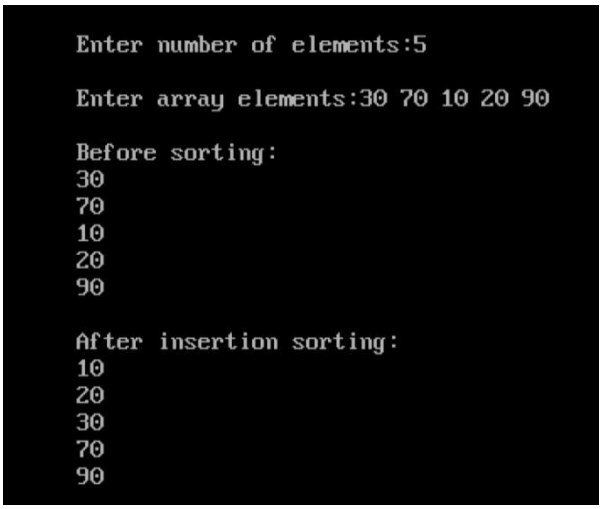
a[j]=a[j+1];

a[j+1]=temp;

}}}

Step 6:Stop

Output:



10. Write a program to sort the given n-elements using selection sort

```
#include<stdio.h>
#include<conio.h>
void selectionsort(int a[],int n);
void main()
  {
            int a[10],i,n;
            clrscr();
            printf("\nEnter the no. of elements:");
           scanf("%d", &n);
           printf("\nEnter the array elements:");
           for(i=0;i<n;i++)
           scanf("%d", &a[i]);
           selectionsort(a, n);
           printf("\nAfter sorting :\n");
           for(i=0;i<n;i++)
           printf("\n %d", a[i]);
           getch();
 }
void selectionsort(int a[], int n)
{
   int i,j,temp, small, pos;
   for(i=1;i<n;i++)
    {
      small=a[i-1];
```

```
    pos=i-1;
    for(j=i;j<n;j++)
     {
        if(a[j]<small)
          {
             small= a[j];
             pos=j;
           }
      }
    temp=a[i-1];
    a[i-1]=a[pos];
    a[pos]=temp;
  }
 }
```

Algorithm for main() function

Step1: Start
Step2: [Input] n
Step3: [repeat] for(i=0;i<n;i++) do
[Read] a[i] array elements
[end for]
Step4: [function call] printarr(a,n)
Step5: [function call] selection(a,n)
Step6: [function call] printarr(a,n)
Step7: Stop

Algorithm for printarr() function

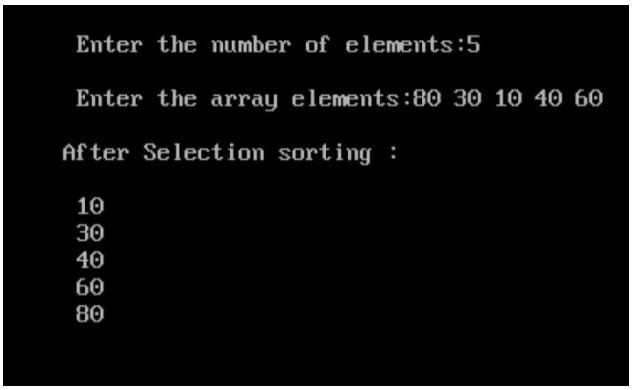
Step1: [repeat] for(i=0;i<n;i++) do
Step2: [output] a[i]
[end for]
Step3: return

Algorithm for Selection() function

Step1: [repeat] for(i=0;i<n-1;i++) do
Small=i
Step2: [repeat] for(j=i+1;j<n;j++) do

[check] if(arr[j]<arr[small]) then
Small=j
[end if]
[end for]
Step3: temp=arr[small]
arr[small]=arr[i]
arr[i]=temp
[end for]
Step4: return

Output:



11. Write a program to sort the given array elements using Merge sort

```
#include<stdio.h>
#include<conio.h>
void mergesort(int a[],int low,int high);
void merge(int a[],int low,int mid,int high);
void main()
{
    int a[10],n,i;
    clrscr();
    printf("\n Enter number of array elements:");
    scanf("%d", &n);
    printf("\n Enter array elements:");
    for(i=0;i<n;i++)
    scanf("\n %d",&a[i]);
    printf("\n \n Before sorting:");
    for(i=0;i<n;i++)
    printf("\n %d",a[i]);
    mergesort(a,0,n-1);
    printf("\n\n After sorting:");
    for(i=0;i<n;i++)
    printf("\n %d",a[i]);
    getch();
}
```

```
void mergesort(int a[],int low,int high)
 {
    int mid;
    if(low<high)
     {
          mid=(low+high)/2;
          mergesort(a,low,mid);
          mergesort(a,mid+1,high);
          merge(a,low,mid,high);
      }
}
void merge(int a[],int low,int mid,int high)
  {
       int i,j,k,c[10];
       i=low;
       j=mid+1;
       k=low;
       while((i<=mid)&&(j<=high))
         {
            if(a[i]<a[j])
              {
                 c[k]=a[i];
                  i++;
                  k++;
              }
```

```
    else
      {
         c[k]=a[j];
         j++;
         k++;
       }
}
 while(i<=mid)
 {
       c[k]=a[i];
       i++;
       k++;
 }
 while(j<=high)
 {
        c[k]=a[j];
        j++;
        k++;
 }
 for(i=low;i<k;i++)
 a[i]=c[i];
 }
```

Algorithm for main() function

Step1: [start]
Step2: [Input] size
Step3: [read] list[i] array elements
Step4: [function call] partition(list,0,size-1)
Step5: [output] list[i] after merge sort
Step6: Stop

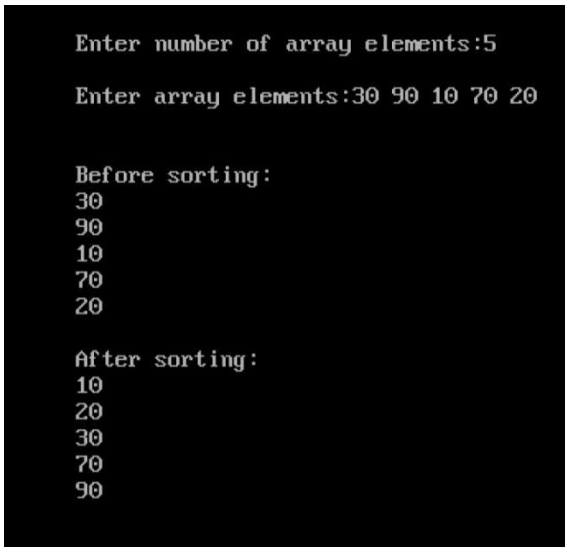
Algorithm for partition() function

Step1: [check] if(low<high) then
mid=(low+high)/2
Step2: [function call] partition(list,low,mid)
Step3: [function call] partition(list,mid+1,high)
Step4: [function call] mergesort(list,low,mid,high)
[end if]
Step5: return

Algorithm for mergesort() function

Step1: [initialize] lo=low
i=low
mi=mid+1
Step2: [Loop] while(lo<=mid)&&(mi<=high) do
[check] if(list[lo]<=list[mi]) then
temp[i]=list[0]
lo=lo+1
[else]
temp[i]=list[mi]
mi=mi+1
[end if]
i=i+1
[end while]
Step3: [check] if(lo>mid) then
[repeat] for(k=mi;k<=high;k++) do
temp[i]=list[k]
i=i+1
[end for]
[else]
[repeat] for(k=lo;k<=mid;k++) do
temp[i]=list[k]
i=i++
[end for]
[end if]
Step4: [repeat] for(k=low;k<=high;k++) do
list[k]=temp[k]
[end for]
Step5: return

Output:


Enter number of array elements:5
Enter array elements:30 90 10 70 20
Before sorting:
30
90
10
70
20
After sorting:
10
20
30
70
90

12. Write a program to sort the given n-elements using Quick sort.

```
#include<stdio.h>
#include<conio.h>
void quicksort(int a[],int low,int high);
void main()
{
    int a[10],n,i;
    clrscr();
    printf("\n Enter number of elements:");
    scanf("%d",&n);
    printf("\n Enter array elements:");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    printf("\n Before quicksort: \n");
    for(i=0;i<n;i++)
    printf(" %d\n ",a[i]);
    quicksort(a,0,n-1);
    printf("\n After quicksort: \n");
    for(i=0;i<n;i++)
    printf(" %d\n ",a[i]);
    getch();
}
void quicksort(int a[],int low,int high)
{
   int i,j,piv,temp;
   i=low+1;
   j=high;
   piv=a[low];
   if(low<high)
   {
```

```
  while(low<high)
  {
   while((a[i]<piv)&&(i<high))
      i++;
   while((a[j]>piv)&&(j>low))
      j--;
   if(i<j)
   {
      temp=a[i];
      a[i]=a[j];
      a[j]=temp;
   }
   else
   {
     temp=a[low];
     a[low]=a[j];
     a[j]=piv;
     break;
   }
  }
   quicksort(a,low,j-1);
   quicksort(a,j+1,high);
   }
}
```

Algorithm:

Step 1 - Consider the first element of the list as **pivot** (i.e., Element at first position in the list).

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

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

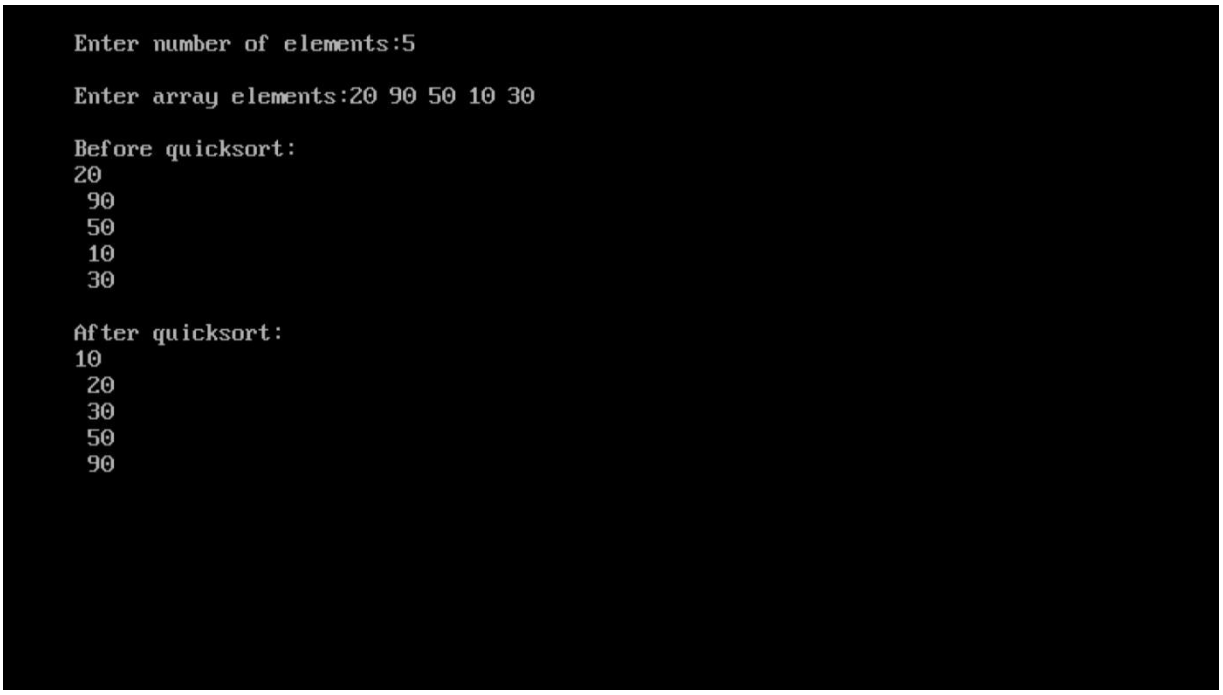
Step 4 - Decrement j until list[j] < pivot then stop.

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

Step 6 - Repeat steps 3,4 & 5 until i> j.

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

Output:



13. Write a program to implement Stacks Operations.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define max 5
int top=-1;
int stack[5];
void Push();
void Pop();
void display();

int main()
{
    int ch;
    clrscr();
    while(1)
    {
        printf("\n\n 1. PUSH");
        printf("\n 2. POP");
        printf("\n 3. DISPLAY");
        printf("\n 4. EXIT");
        printf("\n\nEnter the choice:");
```

```
            scanf("%d",&ch);
            switch(ch)
              {
                    case 1: Push();
                    break;
                    case 2: Pop();
                    break;
                    case 3: display();
                    break;
                    case 4: exit(0);
                    default: printf("\nInvalid choice!!");
              }
        }
}

void Push()
   {
      int ele;
      if(top==max-1)
       {
          printf("\nStack is Overflow!!");
       }
```

```
      else
      {
          printf("\nEnter element to push:");
          scanf("%d",&ele);
          top++;
          stack[top]=ele;
      }
}

void Pop()
 {
   if(top==-1)
    {
      printf("\nStack is Underflow!!");
    }
  else
    {
      printf("\nPopped element:  %d",stack[top]);
      top--;
    }
 }
```

```
void display()
 {
    int i;
    if(top==-1)
     {
        printf("\nStack is Underflow!!");
     }
    else
     {
        printf("\n Top-> ");
        for(i=top;i>=0;i--)
        printf("  %d",stack[i]);
     }
 }
```

Algorithm stack

1. start
2. input n (size of stack)

3 repeat step4

4. switch 1. call push()
   2. call pop()
   3. call display()
   4. exit
5. stop

Algorithm function push()

start
   if top = n then stack full
   top = top + 1
   stack (top) : = item;
Return

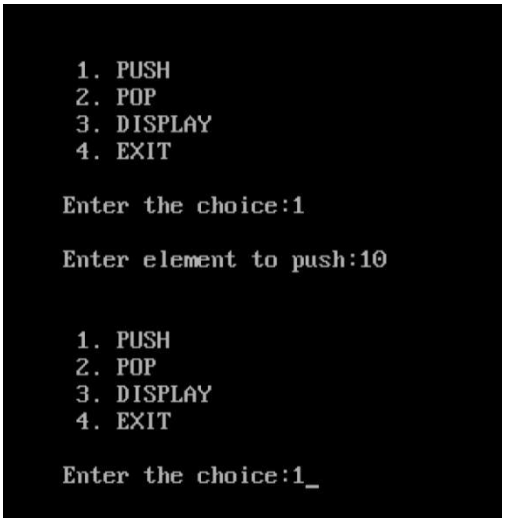
Algorithm function pop()

```
start
   if top <=-1 then stack empty;
   item := stack(top);
   top = top - 1;

  Return
```

Algorithm function display()

```
start
   if top >=0    then
         for (i=top;i>=0;i--)
                  print stack[i];
else
   print "stackempty"
return
```

Output:



```
Enter the choice:1

Enter element to push:20


 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:1

Enter element to push:30


 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:1_
```

```
Enter the choice:1

Enter element to push:40


 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:1

Enter element to push:50
```

```
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:3

 Top->   50  40  30  20  10

 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:2

Popped element:  50
```

```
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:2

Popped element:  40

 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:2

Popped element:  30
```

```
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:      2

Popped element:  20

 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:2

Popped element:  10
```

```
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:2

Stack is Underflow!!

 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT

Enter the choice:3

Stack is Underflow!!
```

14. Write a program to convert Infix expression to Postfix Notation.

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

char stack[20];
int top=-1;
char pop();
void push(char);
int pre(char c) ;

void main()
{
char infix[20],postfix[20];
int i=0,j=0;
char ch;
clrscr();
printf("\n enter infix expression(At end #):");
scanf("%s",&infix);
push('#');
```

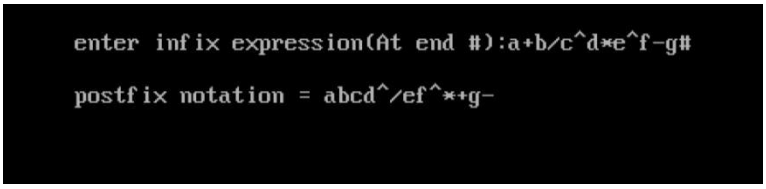
```
while((ch=infix[i++])!='#')
  {
     if(isalnum(ch))
     postfix[j++]=ch;
     else
      {
          while(pre(stack[top])>=pre(ch))
          postfix[j++]=pop();
          push(ch);
       }
    }
while(stack[top]!='#')
   postfix[j++]=pop();
   postfix[j]='\0';
    printf("\n postfix notation = %s ",postfix);
    getch();
}

int pre(char c)
{
   switch (c)
   {
```

```
        case '+':
        case '-':  return (1);
        case '*':
        case '/':  return (2);
        case '^':  return (3);
       case '#': return (0);
    }
}
void push(char c)
 {
    top++;
    stack[top]=c;
 }

char pop()
  {
      char c;
      c=stack[top];
      top--;
      return(c);
  }
```

Algorithm:

Step 1: Start
Step 2: Initialize ans='y' or 'Y'.
Step 3: repeat all steps from 4 to 13 until ans='y ' or 'Y'.
Step 4: read infix expression.
Step 5: Scan the infix expression from left to right.
Step 6: If the scanned character is an operand, output it.
else
Step 7: If the scanned operator is an operator, and stack is empty or contains '(', ')' symbol, push operator into Stack.
Step 8: If the scanned operator has higher precedence than the existing precedence operator in the Stack or if the stack is empty, put it on the stack.
Step 9: If the scanned character has lower precedence than the existing operator in the stack, pop all the Stack operator. After that, push the scanned operator into the stack.
Step 10: If the scanned character is left bracket '(', push it into the stack.
Step 11: If we encountered right bracket ')', pop the stack and print all output string character until '(' is encountered and discard both the bracket.
Step 12: Print the stack output
Step 13: Pop and output all characters, including the operator, from the stack until it is not empty.
Step 14: stop

Output:



15. Write a program to implement Queue Operations.

```
#include <stdio.h>
#include<conio.h>
#include<process.h>
#define max 5
int queue[5];

void Qinsert();
void Qdelete();
void Qdisplay();
int rear = - 1;
int front = - 1;

int main()
{
        int ch;
        clrscr();
        while (1)
          {
            printf("\n\n 1.Insert\n ");
            printf("2.Delete\n ");
            printf("3.Display\n ");
```

```
            printf("4.Exit\n ");
            printf("\n Enter your choice : ");
            scanf("%d", &ch);
            switch(ch)
             {
                 case 1:
                 Qinsert();
                 break;
                 case 2:
                 Qdelete();
                 break;
                 case 3:
                 Qdisplay();
                 break;
                 case 4:
                 exit(0);
                 default:
                 printf("\nInvalid choice ");
              }
        }
}
```

```
void Qinsert()
  {
      int ele;
      if(rear == max - 1)
      printf("\n Queue is FULL ");
      else
       {
          printf("\n Enter the element to insert : ");
          scanf("%d", &ele);
          if(rear== - 1)
           {
              front = 0;
              rear = 0;
           }
         else
            rear++;
            queue[rear] = ele;
       }
  }
```

```
void Qdelete()
  {
     if(front == -1)
     printf("\n Queue is Empty\n");
     else
      {
         printf("\n Element deleted from queue is : %d", queue[front]);
         if(front==rear)
           {
             front=-1;
              rear=-1;
           }
          else
            front++;
       }
   }

void Qdisplay()
  {
     int i;
     if(front == - 1)
     printf("\n Queue is empty\n ");
```

```
    else
     {
        printf("\n Queue is : ");
        printf(" front->");
        for(i = front; i <= rear; i++)
        printf(" %d ", queue[i]);
        printf(" rear");
     }
  }
```

Algorithm:

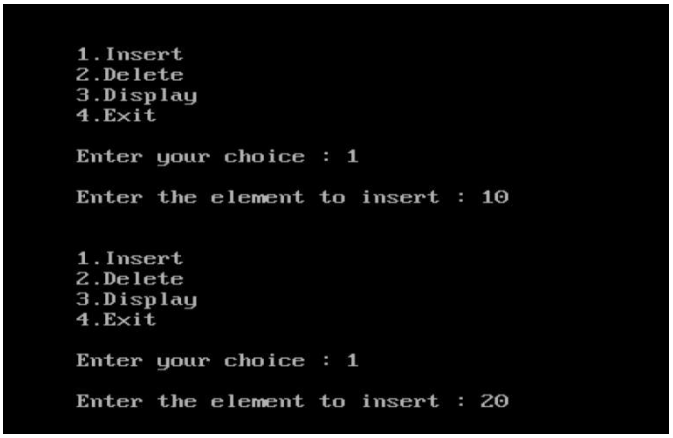
En-queue

Step 1 − Check if the queue is full.
Step 2 − If the queue is full, produce overflow error and exit.
Step 3 − If the queue is not full, increment **rear** pointer to point the next empty space.
Step 4 − Add data element to the queue location, where the rear is pointing.
Step 5 − return success.

De-queue

Step 1 − Check if the queue is empty.
Step 2 − If the queue is empty, produce underflow error and exit.
Step 3 − If the queue is not empty, access the data where front is pointing.
Step 4 − Increment front pointer to point to the next available data element.
Step 5 − Return success

Output:



```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 1

Enter the element to insert : 30


1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 1

Enter the element to insert : 40
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 1

Enter the element to insert : 50


1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 3

Queue is :   front->  10   20   30   40   50   rear
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Element deleted from queue is : 10
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Element deleted from queue is : 20

1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Element deleted from queue is : 30
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Element deleted from queue is : 40

1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Element deleted from queue is : 50
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 2

Queue is Empty
```

```
1.Insert
2.Delete
3.Display
4.Exit

Enter your choice : 3

Queue is empty
```

16. Write a program to Create & Display Singly Linked List and Perform Insertion and Deletion Operations.

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<process.h>

struct node
 {
    int info;
    struct node *next;
 };

struct node *start =NULL;
void create();
void display();
void  iob();
void ioe();
void iom();
void dob();
void dom();
void doe();
```

```
void main()
 {
  int ch;
  clrscr();
  printf("\n create SLL");
  create();
  while(1)
   {
    printf("\n 1. Insert at Begin");
    printf("\n 2. Insert at Middle");
    printf("\n 3. Insert at End");
    printf("\n 4. Delete at Begin");
    printf("\n 5. Delete at Middle");
    printf("\n 6. Delete at End");
    printf("\n 7. Display");
    printf("\n 8. exit");
    printf("\n Enter your choice:");
    scanf("%d", &ch);
    switch(ch)
     {
         case 1: iob();break;
         case 2: iom();break;
```

```
      case 3: ioe();break;
      case 4: dob();break;
      case 5: dob();break;
      case 6: dob();break;
      case 7: display();break;
     case 8: exit(0);
     default : printf("\n invalid choice");
    }
  }
}

void create()
 {
  struct node *temp,*newn;
  int ele;
  char ch;

  do
   {
    printf("\n enter element to insert:");
    scanf("%d",&ele);
```

```
    newn=(struct node *)malloc(sizeof(struct node));
    newn-> info= ele;
    newn-> next=NULL;

    if(start==NULL)
    {
       start=newn;
       temp= newn;
    }
    else
    {
       temp->next=newn;
       temp=newn;
    }
    printf("\n Do you want to insert another node (Y/N):");
    scanf("%s",&ch);
  }while(ch=='Y');
}
void display()
{
  struct node *temp=start;
  if(start==NULL)
```

```
  printf("\n empty singly linked list");
  else
   {
    printf("\nstart->");
    while(temp!=NULL)
     {
        printf("%d->", temp->info);
         temp=temp->next;
     }
    printf("NULL");
   }
}

void iob()
 {
  struct node *newn;
  int ele;
  printf("\n enter element to insert:");
  scanf("%d",&ele);
  newn=(struct node *)malloc(sizeof(struct node));
    newn-> info= ele;
    newn-> next=start;
```

```
    start=newn;
 }

void ioe()
 {
  struct node *newn,*temp=start;
  int ele;
  printf("\n enter element to insert:");
  scanf("%d",&ele);
  newn=(struct node *)malloc(sizeof(struct node));
    newn-> info= ele;
    newn-> next=NULL;
    while(temp->next!=NULL)
     {
      temp=temp->next;
     }
    temp->next=newn;
 }

void iom()
 {
  struct node *newn,*temp1=start, *temp2=start->next;
```

```
int ele,key,flag=0;
  printf("\n enter key node to insert:");
  scanf("%d",&key);
  while(temp1 !=NULL)
   {
    if(temp1-> info==key)
     {
      printf("\n enter element to insert:");
      scanf("%d",&ele);
      newn=(struct node *)malloc(sizeof(struct node));
      newn-> info= ele;
      temp1->next=newn;
      newn->next=temp2;
      flag=1;
      break;
     }
    else
    {
      temp1=temp1->next;
      temp2=temp2->next;
    }
 }
```

```
    if(flag==0)
       printf("\n key element not found");
    else
       printf("\n key element is inserted successfully");
 }

void dob()
 {
  struct node *temp=start;
  if(start==NULL)
  printf("\n empty SLL");
  else
   {
       printf("\n deleted node =%d", temp->info);
       start = temp->next;
       free(temp);
   }
}

void doe()
 {
   struct node*temp1=start,*temp2=start->next;
```

```
    if(start==NULL)
    printf("\n empty SLL");
    else
     {
       while(temp2-> next!=NULL)
        {
          temp1=temp1->next;
          temp2=temp2->next;
        }
          printf("\n deleteed node=%d",temp2->info);
          temp1->next=NULL;
          free(temp2);
       }
}

void dom()
{
    struct node *temp1=start,*temp2=start->next;
    int key,flag=0;
    if(start==NULL)
    printf("\n empty SLL");
    else
```

```
    {
       printf("\n enter key node to delete:");
       scanf("%d",&key);
       while(temp2!=NULL)
       if(key==temp2->info)
        {
           printf("\n deleted node=%d", temp2->info);
           temp1->next=temp2->next;
           free(temp2);
           flag=1;
           break;
         }
        else
         {
            temp1=temp1->next;
            temp2=temp2->next;
          }
    }
  if(flag==0)
      printf("\n key node not found");
}
```

Algorithm for Create Linked list.

Step1: start
Step2:Structure variables *temp,*ptr.
Step3: temp=(struct node *)malloc (size of (struct node))
Step 4: if(temp==NULL)
Print “out of memory space”
Exit.
Input data
Read temp->info
Temp->next=NULL
Step5: if(start==NULL)
Start=temp
Else
Ptr=start
While(ptr->next=NULL)
Ptr=ptr->next
Ptr->next=temp

Algorithm for Display Elements in Linked list.

Step1: start
Step2: Structure variables *temp,*ptr.
Step3: if(start==NULL)
Print “List Empty”
Else
Ptr=start
While(ptr!=NULL)
Print ptr->info
Ptr=ptr->next
Step4:stop.

Algorithm for Insert Node at Beginning of Linked list.

Step1: start
Step2:Structure variables *temp.
Step3: temp=(struct node *)malloc (size of (struct node))
Step 4: if(temp==NULL)
Print “out of memory space”
Exit.
Input data
Read temp->info
Temp->next=NULL
Step5:if(start==NULL)
Start=temp
Else
Temp->next=start
Start=temp
Step6: stop.

Algorithm for Deleting a node at Beginning of Linked list.

Step1: start
Step2: Structure variables *ptr.
Step3: if(start==NULL)
Print “List Empty”
Else
Ptr=start
Start=start->next
Print “deleted element is “ ptr->info
Free(ptr)
Step4:stop.

Output:

```
create SLL
enter element to insert:10

Do you want to insert another node (Y/N):Y

enter element to insert:20

Do you want to insert another node (Y/N):N
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:7

start->10->20->NULL
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:1

enter element to insert:100
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:3

enter element to insert:200

1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:2_
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:2

enter key node to insert:10

enter element to insert:60

key element is inserted successfully
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:7

start->100->10->60->20->200->NULL

1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:3
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:3

enter element to insert:90
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:7

start->100->10->60->20->200->NULL
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:4

deleted node =100
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:6_
```

```
deleted node =10
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:5

deleted node =60
```

```
1. Insert at Begin
2. Insert at Middle
3. Insert at End
4. Delete at Begin
5. Delete at Middle
6. Delete at End
7. Display
8. exit
Enter your choice:7

start->20->200->NULL
```

17. Write a program to Create binary search tree and apply tree traversal technique.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<alloc.h>

struct node
 {
   struct node *lchild;
   int info;
   struct node *rchild;
 };
struct node *root =NULL;
void create();
void preorder(struct node *temp);
void inorder(struct node *temp);
void postorder(struct node *temp);
void main()
 {
  int ch;
  clrscr();
```

```
 while(1)
 {
   printf("\n 1. Create");
   printf("\n 2. Pre-order");
   printf("\n 3. Inorder");
   printf("\n 4. Post-order");
   printf("\n 5. exit");
   printf("\n Enter your choice:");
   scanf("%d", &ch);
   switch(ch)
    {
        case 1: create();break;
        case 2: preorder(root);break;
        case 3: inorder(root);break;
        case 4: postorder(root);break;
        case 5: exit(0);
        default : printf("\n invalid choice");
    }
  }
}
void create()
 {
```

```
struct node *newn ,*parent,*temp=root;
int ele;
printf("\n enter element to insert:");
scanf("%d",&ele);
newn=(struct node *)malloc(sizeof(struct node));
newn-> info= ele;
newn-> lchild=NULL;
newn-> rchild=NULL;
  if(root==NULL)
     root=newn;
  else
 {
   while(temp!=NULL)
    {
       parent=temp;
       if(newn->info < temp->info)
          temp=temp->lchild;
       else
          if(newn->info > temp->info)
       temp=temp->rchild;
       else
       break;
```

```
        }
    if(newn->info < parent->info)
    parent->lchild=newn;
    else
    if(newn->info>parent->info)
        parent->rchild=newn;
    else
        printf("\n Node is already inserted (search is successful)");
  }
}

void preorder(struct node *temp)
{
  if (temp!=NULL)
  {
    printf("%d\n",temp->info);
    preorder(temp->lchild);
    preorder(temp->rchild);
  }
}

void inorder(struct node *temp)
```

```
{
  if (temp!=NULL)
  {
    inorder(temp->lchild);
    printf("%d\n",temp->info);
    inorder(temp->rchild);
  }
}

void postorder(struct node *temp)
{
  if (temp!=NULL)
  {
    postorder(temp->lchild);
    postorder(temp->rchild);
    printf("%d\n",temp->info);

  }
}
```

Algoritm:

1. First create a new node
Define a node having some data, references to its left and right child nodes.
2. Make the first value that you insert as root.
3. If you want to insert the next value to the left of root, insert as root->left, or if you want to insert to right insert as root->right

4. Repeat step 3 to insert the node in the right position in the Tree. For example, to insert the node to the left left of the root, insert as root->left->left.
5. traverse tree in inorder.
6. traverse tree in preorder.
7. traverse tree in postorder.
8. stop

Algorithm Inorder(tree)

1. Traverse the left subtree, i.e., call Inorder(left-subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right-subtree)

Algorithm Preorder(tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left-subtree)
3. Traverse the right subtree, i.e., call Preorder(right-subtree)

Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.

18. Write a program to accept cities name and display it in alphabetical order.

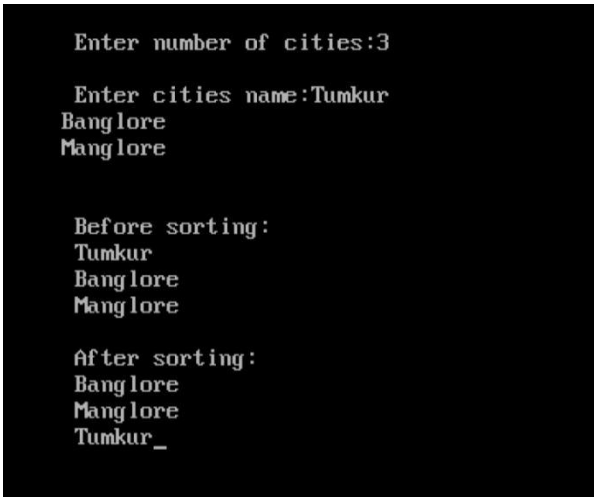
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char city[10][10],temp[10];
        int n,i,j;
        clrscr();
        printf("\n Enter number of cities:");
        scanf("%d",&n);
        printf("\n Enter cities name:");
        for(i=0;i<n;i++)
                scanf("%s",city[i]);
        printf("\n\n Before sorting:");
        for(i=0;i<n;i++)

                printf("\n %s",city[i]);
                for(i=0;i<n-1;i++)
                {

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

```
                {
                        if(strcmp(city[i],city[j])>0)
                        {
                                strcpy(temp,city[i]);
                                strcpy(city[i],city[j]);
                                strcpy(city[j],temp);
                        }
                }
        }
        printf("\n\n After sorting:");
        for(i=0;i<n;i++)
                printf("\n %s",city[i]);
        getch();
 }
```

ALOGRITHM

STEP1:[START]
Program to read name of cities and arrange in alphabetically
STEP2[INPUT]
N,str[i]
STEP3;[CONDITION]
For(i=0;i<n;i++)
  For(j=i+1;j<n;j++)
    If(strcmp(str[i],str[i]>0)
Strcpy(s,str[i])
Strcpy(str[i],str[j])
Strcpy[i],s)
STEP4:[OUTPUT]
  Str[i]
STEP5:[END]
  STOP

Ouput:


Enter number of cities:3
Enter cities name:Tumkur
Banglore
Manglore
Before sorting:
Tumkur
Banglore
Manglore
After sorting:
Banglore
Manglore
Tumkur_