PgmNo.1

Date:

REVERSE A STRING USING POINTERS.

……..

Aim: Write a program to reverse a string using pointers

ALGORITHM:

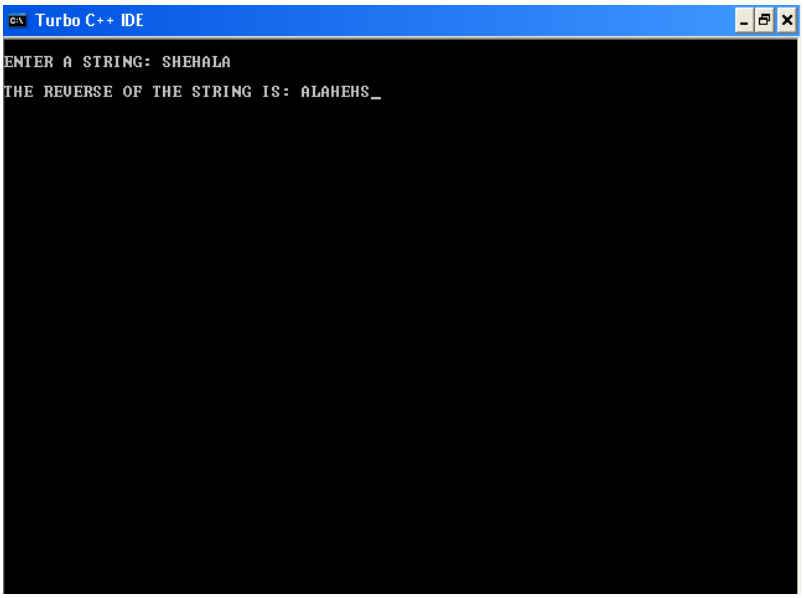
STEP 1: Start.
STEP 2: Declare len,i, as integer
STEP 3:Declare*s as character.
STEP 4:Read the string
STEP5: Set len=length of the string
STEP6: set i=len
STEP 7: Repeat step 7 to STEP 8 until i>=0
STEP 8: print the value of *(s+i)
STEP 9: Goto STEP 6
STEP 10: STOP.

Program:

```
#include  <stdio.h>
#include <conio.h>
#include<string.h>
void main()
{
        char *s;
        intlen,i;
        clrscr();
        printf("\nENTER A STRING: ");
        gets(s);
        len=strlen(s);
        printf("\nTHE REVERSE OF THE STRING IS:");
        for(i=len;i>=0;i--)
        printf("%c",*(s+i));
        getch();
}
```

Output:

Sample output



Pgm No.2

Date:

IMPLEMENT PATTERN MATCHING ALGORITHM.

……..

Aim: Write a program to Implement Pattern matching algorithm

ALGORITHM:

STEP 1: Start.
STEP 2: Declare text[100],pattern[100] as character array and count=0 as character variable.
STEP 3:Declare text_leng,pattern_leng,max,i,j,k as integer variable.
STEP4:Read the value of text
STEP5: Read the value of pattern
STEP6: set text_leng=strlen(text)
STEP 7: Set pattern_leng=strlen(pattern)
STEP 8: Check pattern_leng>text_leng. Go to step 9 else goto step 10
STEP 9:Pattern not found
STEP 10:set max= text_leng-pattern_leng+1
STEP 11: Repeat step 12 to step15 until i<max
STEP 12: Repeat step 13 to STEP 15 until k<pattern_leng
STEP 13:Check pattern[k]==text[i+k] goto step 14 else goto15
STEP 14:set count=count+1;
STEP 15: break
STEP 16:check count== pattern_leng goto step 17 else goto 18
STEP 17: Print pattern is found at position i
STEP 18: Print pattern not found
STEP 19:Stop

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
Void  main()
{
        char text[100],pattern[100],count=0;
        int text_leng,pattern_leng,max,i,j,k;
        clrscr();
        printf("enter the text:");
        gets(text);
        printf("enter the pattern:");
        gets(pattern);
        text_leng=strlen(text);
        pattern_leng=strlen(pattern);
        if(pattern_leng >text_leng)
        {
                printf("pattern is not found ");
        }
        max=text_leng-pattern_leng+1;
```

```
        for(i=0;i<max;i++)
        {

                for(k=0;k<pattern_leng;k++)
                {
                        if(pattern[k]==text[i+k])
                        {
                                count++;
                        }
                        else
                        {
                        break;
                        }
                }
        }
        if(count==pattern_leng)
        {
        printf("pattern is found at%d position",i);
        }
        else
        {
        printf("pattern not found");
        }
        getch();
}
```

Output:

Sample output:

Pgm No.3

Date:

SEARCH AN ELEMENT IN THE 2 D ARRAY

……..

Aim: Write a program to search an element in the 2 d array

ALGORITHM:

STEP 1: Start.
STEP 2: Declare I,j,row,col,val,found as integer and a as a 2D integer array.
STEP 3:Read number of rows and columns in row and col
STEP4:Read the 2D array elements
STEP5: Print the array elements
STEP6: Read the element to be searched in val
STEP 7: Repeat step 8 to step11 until i<row
STEP 8: Repeat step 9 to STEP 11 until j<col
STEP 9:Check a[i][j]==val goto step 10 else goto12
STEP 10:Print the item found at position
STEP 11: set found=1
STEP 12:check found==0 goto step 13
STEP 13:Print the item is not found
STEP 14:Stop

Program:

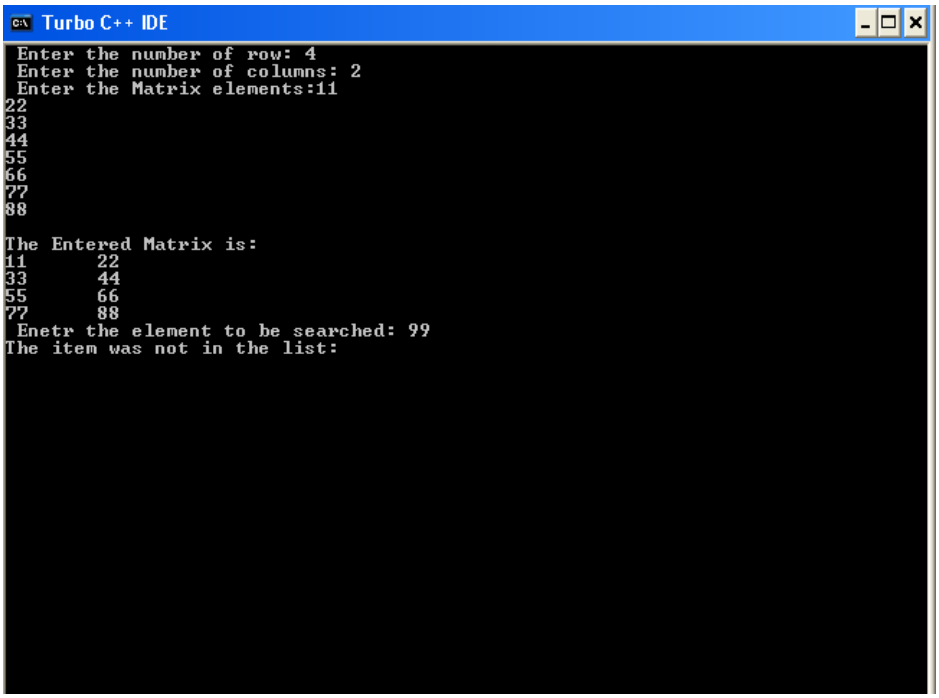
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10][10],i, j, row, col, val,found=0;
        clrscr();
        printf(" Enter the number of row: ");
        scanf("%d", &row);
        printf(" Enter the number of columns: ");
        scanf("%d", &col);
        printf(" Enter the Matrix elements:");
        for(i=0; i<row; i++)
 {
  for(j=0; j<col; j++)
  {
```

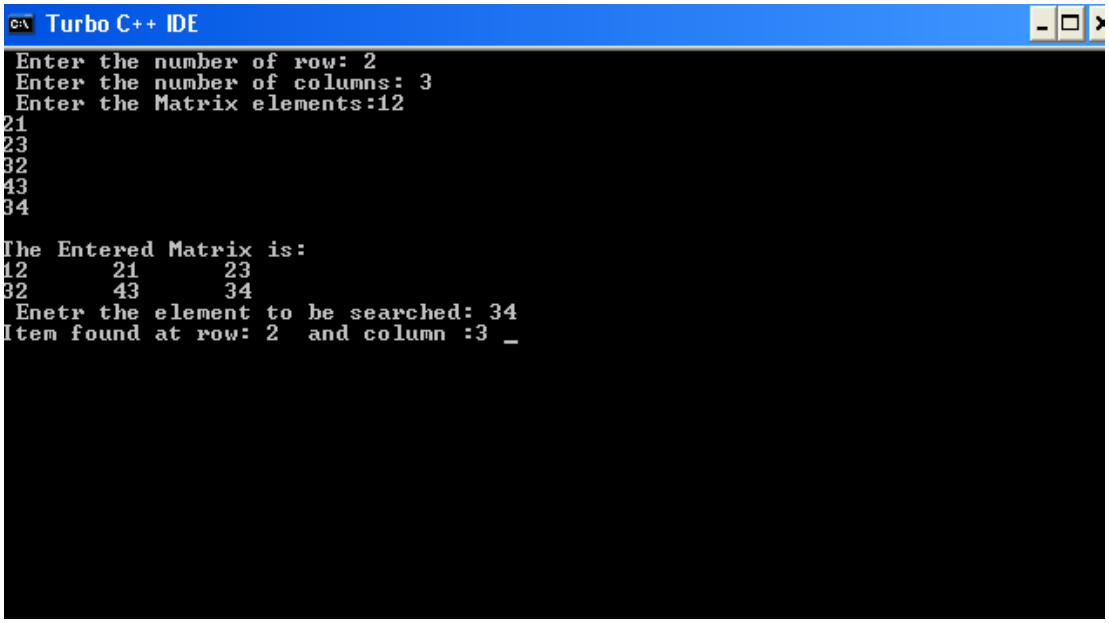
```
    scanf("%d", &a[i][j]);
   }
  }
 printf(" \nThe Entered Matrix is:\n");
 for(i=0; i<row; i++)
  {
   for(j=0; j<col; j++)
   {
    printf("%d \t", a[i][j]);
   }
   printf("\n");
  }
 printf(" Enetr the element to be searched: ");
 scanf("%d", &val);
 for(i=0; i<row; i++)
  {
   for(j=0; j<col; j++)
   {
     if(a[i][j]==val)
      {
       printf("Item found at row: %d and column :%d ", i+1,j+1);
       found=1;
      }
   }

  }
 if(found==0)
  {
   printf("The item was not in the list:");
  }
```

```
 getch();
}
```

output:

sample output:





Pgm No.4
Date:

APPEND 2 ARRAYS

……..

Aim: Write a program to Append 2 arrays

ALGORITHM:

STEP 1: Start
STEP 2: Declare a[5],b[5]and c[5] as integer array
STEP 3:Declare i,j,n,m as integer variable.
STEP4:Read the size of first array in n
STEP5: Read the elements of first array in a[i]
STEP6:Read the size of second array in m
STEP7: Read the elements of second array in b[i]
STEP 8: Repeat step 9 until i<n
STEP 9:Set c[i]=a[i]
STEP 10: Repeat step 11 until j,m
STEP 11:set c[i++]=b[j] and j=i
STEP 12:Print the value of c[i]
STEP 11:stop

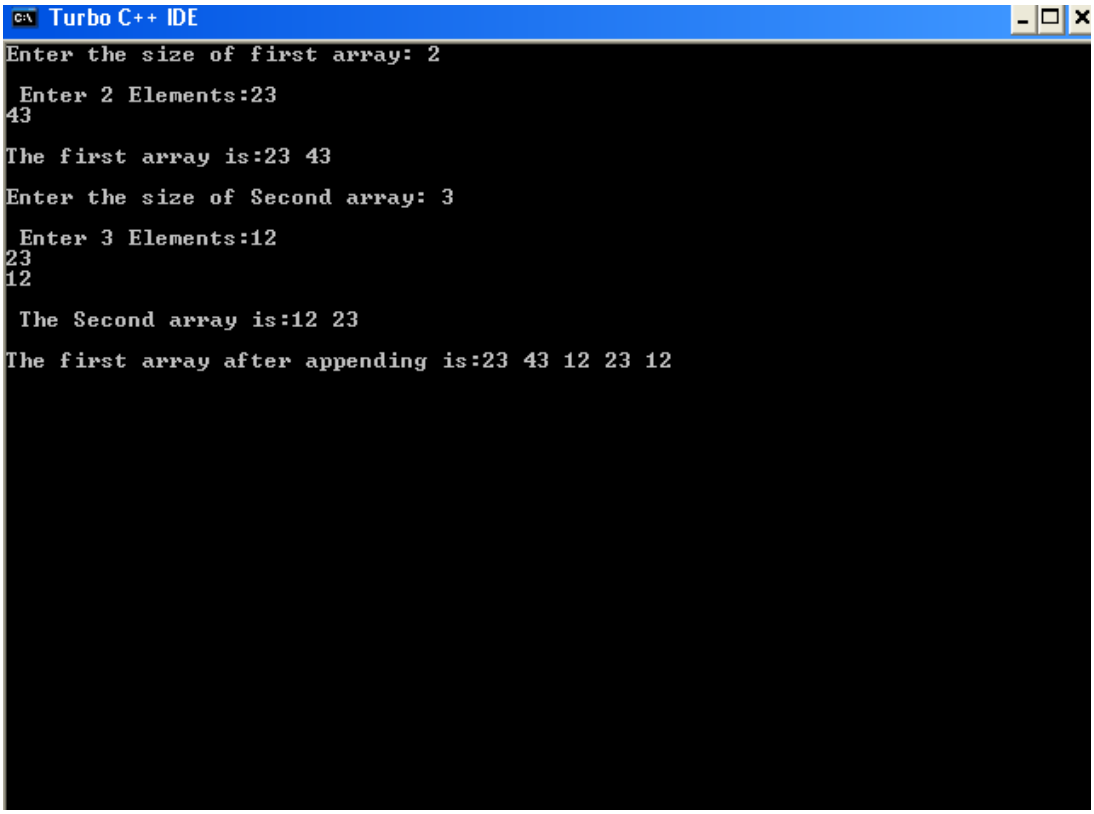
program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[5],b[5],c[10],i,j,n,m ;
        clrscr();
        printf("Enter the limit of the first array");
        scanf("%d",&n);
        printf("Enter the elements of first array");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        printf("Enter the limit of the second array");
        scanf("%d",&m);
        printf("Enter the elements of second array");
        for (i=0;i<m;i++)
        {
        scanf("%d",&b[i]);
        }
        for (i=0;i<n;i++)
        c[i]=a[i];
        for(j=0;j<m;j++)
```

```
        c[i++]=b[j];
        j=i;
        printf("After concatenation:\n");
        for (i=0; i<j; i++)
        {
        printf("%d\n", c[i]);
        }
        getch ( );
}
```

output:

sample output:



Pgm No.5
Date:

BINARY SEARCH

……..

Aim: Write a program to search an element in the array using binary search

ALGORITHM:

STEP 1: Start
STEP 2: Declare a[100] as integer array
STEP 3:Declare c, first, last, middle, n, search as integer variables.
STEP4:Read the size of array in n
STEP5: Read the elements of array in a[]
STEP6:Read the item to be searched in search
STEP7: set first=0,last =n-1,middle=(first+last)/2
STEP 8: Repeat step 9 to step 13 until first <=last
STEP 9:Check a[middle]<search goto step 10 else goto step 11
STEP 10: first=middle+1
STEP 11: Check a[middle]=search goto step 12 else goto step 13
STEP 12:Print the item found at location middle+1
STEP 13:set last=middle-1 and middle=(first+last)/2
STEP 14:check first>last goto step 15
STEP 15:Print the item is not found in the list
STEP 16:Stop

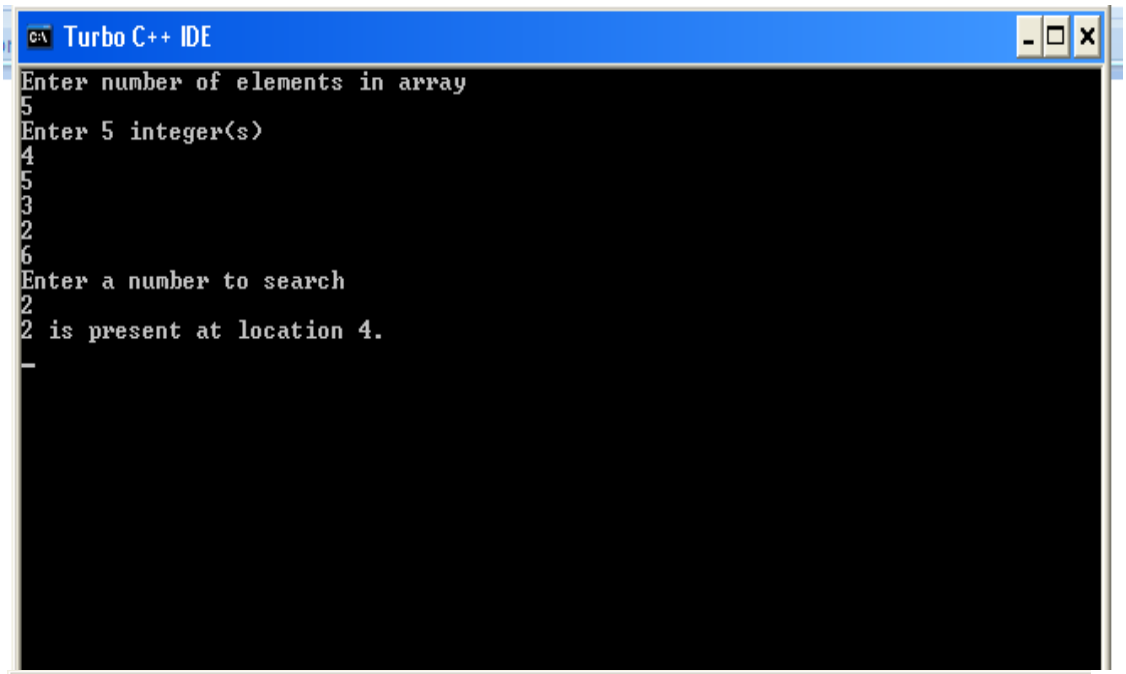
Program:

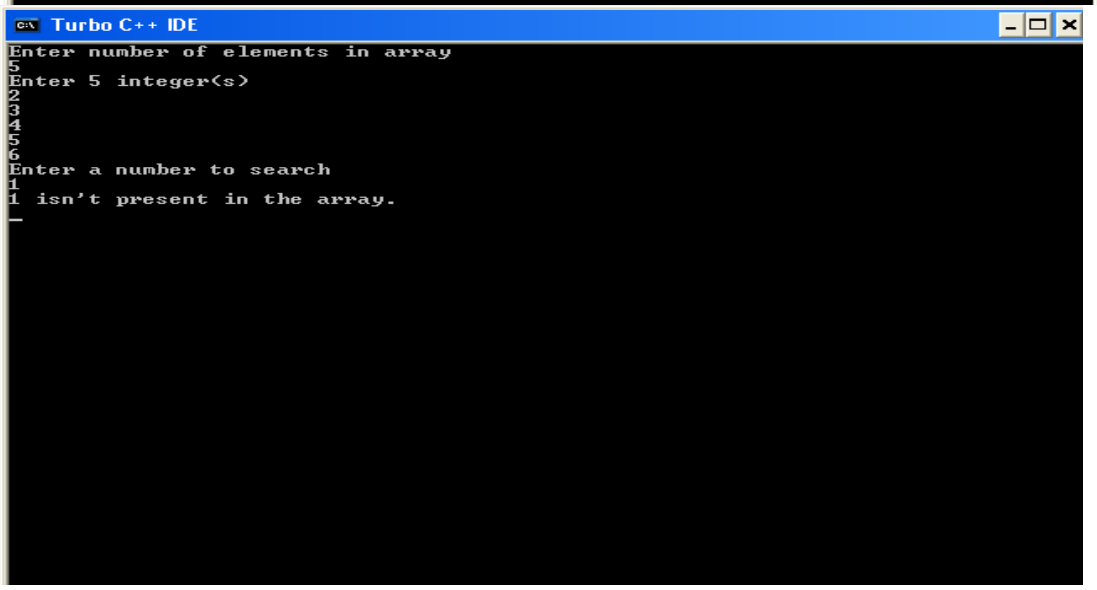
```
#include <stdio.h>

int main()

{

int c, first, last, middle, n, search, a[100];

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter %d integers\n", n);

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

scanf("%d",&a[c]);

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;
```

```
middle = (first+last)/2;
while (first <= last) {
if (a[middle] < search)
first = middle + 1;
else if (a[middle] == search)
 {
printf("%d found at location %d.\n", search, middle+1);
break;
      }
else
last = middle - 1;
middle = (first + last)/2;
   }
if (first > last)
printf("Not found! %d isn't present in the list.\n", search);
return 0;
}
```

output:

sample output:





Pgm No.6
DATE:

SPARSE MATRIX

……..

Aim: Write a program to real a sparse matrix and display its triplet representation using array.

Algorithm:

STEP 1:Start

STEP 2:Declare a sparse matrix of class 5x6 with 6 non-zero values

STEP 3:Finding total non-zero values in the sparse matrix

STEP 4:Defining result Matrixaccording to the condition

a)for(i=0;i<nzero+1;i++)

b)for(j=0;j<3;j++)

STEP 5:Displaying result matrix

STEP 6:Stop

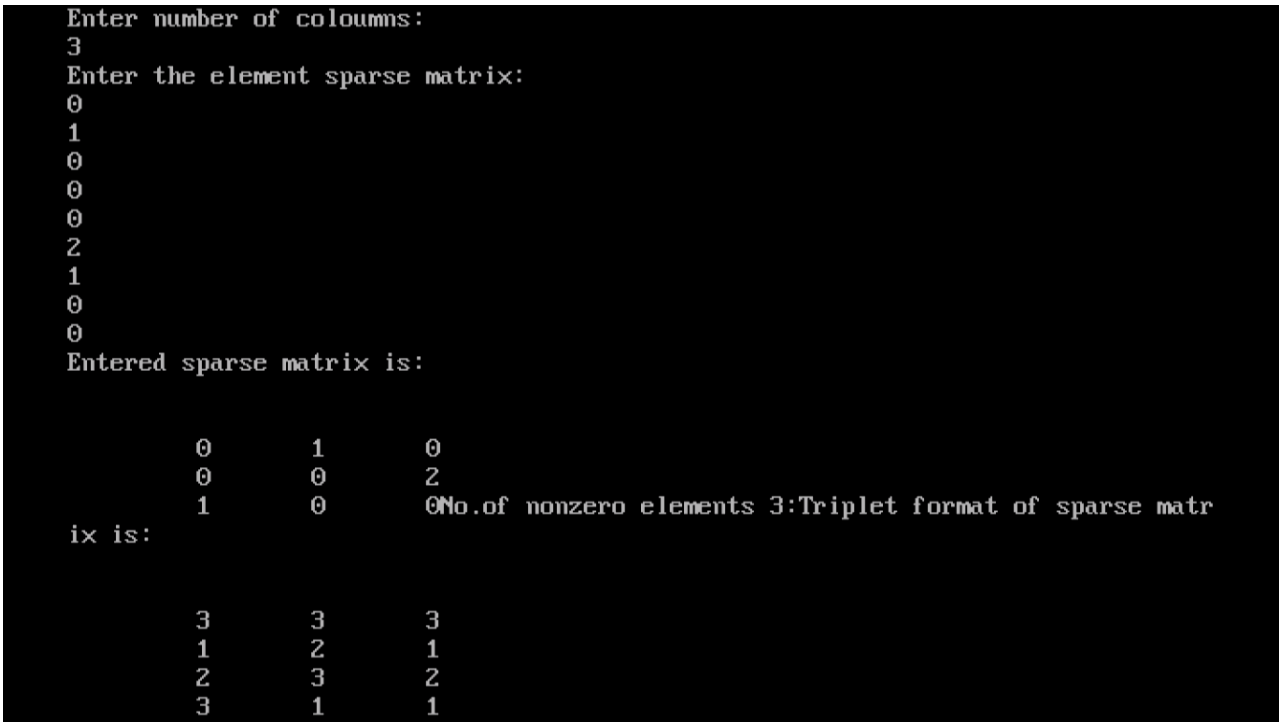
Program:

```
#include<stdio.h>
#include<string.h>
void main()
 {
 int S[5][3],i,j,nzero=0,m,n,k,A[10][10],s;
clrscr();
  printf("\nenter number of rows");
  scanf("%d",&m);
  printf("\nenter number of coloumns");
  scanf("%d",&n );
   printf("\nenter the element sparse matrix");

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

```
  {
for(j=0;j<n;j++)
 {
scanf("%d",&A[i][j]);
 }
}
  printf("\nenter sparse matrix");
for(i=0;i<m;i++)
  {
  printf("\n"); for(j=0;j<n;j++)
 {
  printf("\t %d",A[i][j]);
if(A[i][j]!=0)
 {
nzero++;
 }
 }
}
 printf("\n no.of nonzero elements %d",nzero);
S[0][0]=m;
S[0][1]=n;
S[0][2]=nzero;
k=1;
for(i=0;i<m;i++)
 {
for(j=0;j<n;j++)
```

```
{
if(A[i][j]!=0)
 {
 S[k][0]=i+1;
 S[k][1]=j+1;
 S[k][2]=A[i][j];
 k++;
  }
 }

  printf("\nttriiplet format of sparse matrix");
 for(i=0;i<nzero+1;i++)
 {
 printf("\n");
 for(j=0;j<3;j++)
 {
 printf("\t %d",S[i][j]);
 }
 }
 getch();
 }
```

Output:


Enter number of coloumns:
3
Enter the element sparse matrix:
0
1
0
0
0
2
1
0
0
Entered sparse matrix is:
0 1 0
0 0 2
1 0 0No.of nonzero elements 3:Triplet format of sparse matr
ix is:
3 3 3
1 2 1
2 3 2
3 1 1

Pgm No.7

Date:

SINGLY LINKED – CREATE AND DISPLAY

……..

Aim: Create a singly linked list of n nodes and display it.

Algorithm:

STEP 1:Start

STEP 2:Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list.

STEP 3:Create another class which has two attributes: head and tail.

STEP 4:addNode() will add a new node to the list:

a. Create a new node.
b. It first checks, whether the head is equal to null which means the list is empty.
c. If the list is empty, both head and tail will point to the newly added node.
d. If the list is not empty, the new node will be added to end of the list such that tail's next will point to the newly added node. This new node will become the new tail of the list.

STEP 5:reverse() will reverse the order of the nodes present in the list.

. This method checks whether node next to current is null which implies that, current is pointing to tail, then it will print the data of the tail node.
    a. Recursively call reverse() by considering node next to current node and prints out all the nodes in reverse order starting from the tail.

STEP 6:display() will display the nodes present in the list:

. STEP 7:Define a node current which will initially point to the head of the list.
    a. Traverse through the list till current points to null.
    b. Display each node by making current to point to node next to it in each iteration

STEP 8:Stop

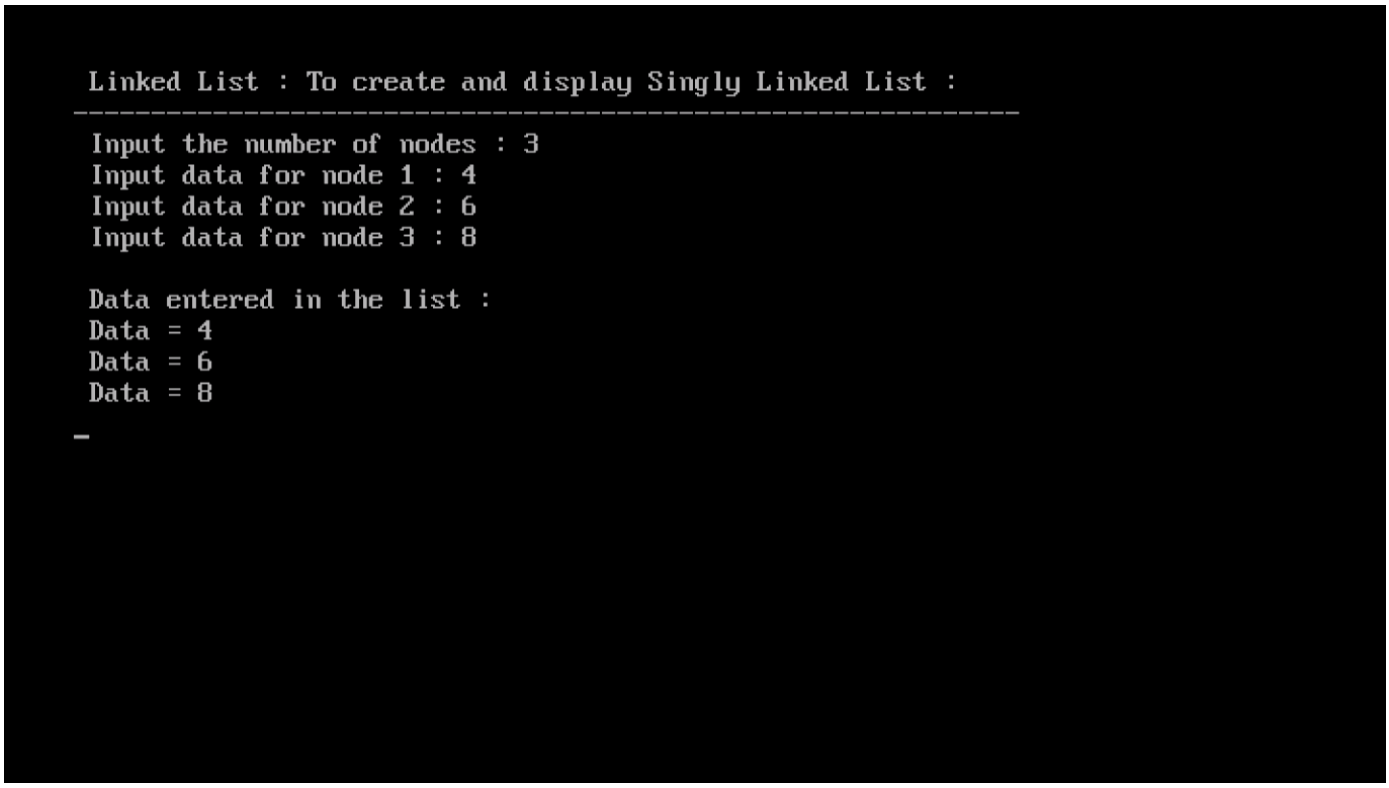
Program:

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

struct node
{
   int num;
   struct node *nextptr;
}*stnode;

void createNodeList(int n);
void displayList();

void main()
{
   int n;
   clrscr();
                printf("\n\n Linked List : To create and display Singly Linked List :\n");
                printf("-------------------------------------------------------------\n");

   printf(" Input the number of nodes : ");
   scanf("%d", &n);
   createNodeList(n);
   printf("\n Data entered in the list : \n");
   displayList();
  // return 0;
  getch();
}
```

```
void createNodeList(int n)
{
   struct node *fnNode, *tmp;
   int num, i;
   stnode = (struct node *)malloc(sizeof(struct node));
   if(stnode == NULL)
   {
      printf(" Memory can not be allocated.");
   }
   else
   {
      printf(" Input data for node 1 : ");
      scanf("%d", &num);
      stnode->num = num;
      stnode->nextptr = NULL;
      tmp = stnode;
      for(i=2; i<=n; i++)
      {
         fnNode = (struct node *)malloc(sizeof(struct node));
         if(fnNode == NULL)
         {
            printf(" Memory can not be allocated.");
            break;
         }
         else
         {
            printf(" Input data for node %d : ", i);
            scanf(" %d", &num);
             fnNode->num = num;
```

```
            fnNode->nextptr = NULL;
             tmp->nextptr = fnNode;
            tmp = tmp->nextptr;
          }
       }
   }
}
void displayList()
{
   struct node *tmp;
   if(stnode == NULL)
   {
      printf(" List is empty.");
   }
   else
   {
      tmp = stnode;
      while(tmp != NULL)
      {
         printf(" Data = %d\n", tmp->num);
         tmp = tmp->nextptr;
      }
   }
}
```

Output:


Linked List : To create and display Singly Linked List :
Input the number of nodes : 3
Input data for node 1 : 4
Input data for node 2 : 6
Input data for node 3 : 8
Data entered in the list :
Data = 4
Data = 6
Data = 8

PgmNo.8

Date:

SINGLY LINKED LIST- DELETION

……..

Aim: Delete a given node from a singly linked list.

Algorithm:

STEP 1:Start
STEP 2: if head = null
write underflow
goto step 10
end of if
STEP 3: set temp = head
STEP 4: set i = 0
step 5: repeat STEP 5 TO 8 UNTIL I<loc< li=""></loc<>
STEP 6: temp1 = temp
STEP 7: temp = temp → next
STEP 8: if temp = null
write "desired node not present"
goto step 12
end of if
STEP 9: i = i+1
end of loop
STEP 10: temp1 → next = temp → next
STEP 11: free temp
STEP 12: stop

Program:

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

struct node
{
   int num;
   struct node *nextptr;
  }*stnode;

void createNodeList(int n);
void FirstNodeDeletion();
   void displayList();

int main()
{
   int n,num,pos;
```

```
                printf("\n\n Linked List : Delete first node of Singly Linked List :\n");
                printf("-------------------------------------------------------------\n");
    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    createNodeList(n);
    printf("\n Data entered in the list are : \n");
    displayList();
    FirstNodeDeletion();
    printf("\n Data, after deletion of first node : \n");
    displayList();
    return 0;
}
void createNodeList(int n)
{
    struct node *fnNode, *tmp;
    int num, i;
    stnode = (struct node *)malloc(sizeof(struct node));
    if(stnode == NULL)
    {
        printf(" Memory can not be allocated.");
    }
    else
    {
// reads data for the node through keyboard
        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode-> num = num;
        stnode-> nextptr = NULL;
        tmp = stnode;
//Creates n nodes and adds to linked list
        for(i=2; i<=n; i++)
        {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode == NULL)                                    {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);
                fnNode->num = num;
                fnNode->nextptr = NULL;
                 tmp->nextptr = fnNode;
                tmp = tmp->nextptr;
            }
        }
    }
```

```
}

void FirstNodeDeletion()
{
   struct node *toDelptr;
   if(stnode == NULL)
   {
      printf(" There are no node in the list.");
   }
   else
   {
      toDelptr = stnode;
      stnode = stnode->nextptr;
      printf("\n Data of node 1 which is being deleted is :  %d\n", toDelptr->num);
      free(toDelptr);
   }
}


void displayList()
{
   struct node *tmp;
   if(stnode == NULL)
   {
      printf(" No data found in the list.");
   }
   else
   {
      tmp = stnode;
      while(tmp != NULL)
      {
         printf(" Data = %d\n", tmp->num);
         tmp = tmp->nextptr;
      }
   }
}
```

Output:

Sample output:

```
 Linked List : Delete first node of Singly Linked List :
------------------------------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 4
 Input data for node 2 : 5
 Input data for node 3 : 6

 Data entered in the list are :
 Data = 4
 Data = 5
 Data = 6

 Data of node 1 which is being deleted is :  4

 Data, after deletion of first node :
 Data = 5
 Data = 6
```

PgmNo.9

Date:

DOUBLY LINKED LIST-FORWARD & BACKWARD DISPLAY

……..

Aim: Create a doubly linked list of integers and display in forward and backward direction.

Algorithm:

STEP 1:Start

STEP 2:Define a Node class which represents a node in the list. It will have three properties: data, previous which will point to the previous node and next which will point to the next node.

STEP 3:Define another class for creating a doubly linked list, and it has two nodes: head and tail. Initially, head and tail will point to null.

STEP 4:addNode() will add node to the list:

a. It first checks whether the head is null, then it will insert the node as the head.
b. Both head and tail will point to a newly added node.
c. Head's previous pointer will point to null and tail's next pointer will point to null.
d. If the head is not null, the new node will be inserted at the end of the list such that new node's previous pointer will point to tail.
e. The new node will become the new tail. Tail's next pointer will point to null.

STEP 5:reverse() will reverse the given doubly linked list.

STEP 6:Define a node current which will initially point to head.

f. Traverse through the list by making current to point to current.next in each iteration till current points to null.
g. In each iteration, swap previous and next pointer of each node to reverse the direction of the list.
h. In the end, swap the position of head and tail.

STEP 7:display() will show all the nodes present in the list.

. STEP 8:Define a new node 'current' that will point to the head.

a. Print current.data till current points to null.

b. Current will point to the next node in the list in each iteration.

STEP 9:Stop

Program:

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

struct node {
   int num;
   struct node * preptr;
   struct node * nextptr;
}*stnode, *ennode;


void DlListcreation(int n);
void displayDlList();
void displayDlListRev();
int main()
{
   int n;
   stnode = NULL;
   ennode = NULL;
        printf("\n\n Doubly Linked List : Create and display a doubly linked list :\n");
        printf("--------------------------------------------------------------------\n");

   printf(" Input the number of nodes : ");
   scanf("%d", &n);

   DlListcreation(n);
   displayDlList();
  displayDlListRev();


   return 0;
}
```

```
void DlListcreation(int n)
{
   int i, num;
   struct node *fnNode;

   if(n >= 1)
   {
      stnode = (struct node *)malloc(sizeof(struct node));

      if(stnode != NULL)
      {
         printf(" Input data for node 1 : "); // assigning data in the first node
         scanf("%d", &num);

         stnode->num = num;
         stnode->preptr = NULL;
         stnode->nextptr = NULL;
         ennode = stnode;
// putting data for rest of the nodes
         for(i=2; i<=n; i++)
         {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode != NULL)
            {
               printf(" Input data for node %d : ", i);
               scanf("%d", &num);
               fnNode->num = num;
               fnNode->preptr = ennode;    // new node is linking with the previous node
               fnNode->nextptr = NULL;

               ennode->nextptr = fnNode;   // previous node is linking with the new node
               ennode = fnNode;            // assign new node as last node
            }
            else
            {
               printf(" Memory can not be allocated.");
               break;
            }
         }
      }
      else
      {
         printf(" Memory can not be allocated.");
      }
   }
}
void displayDlList()
```

```
{
   struct node * tmp;
   int n = 1;
   if(stnode == NULL)
   {
      printf(" No data found in the List yet.");
   }
   else
   {
      tmp = stnode;
      printf("\n\n Data entered on the list are :\n");

      while(tmp != NULL)
      {
         printf(" node %d : %d\n", n, tmp->num);
         n++;
         tmp = tmp->nextptr; // current pointer moves to the next node
      }
   }
}
void displayDlListRev()
{
   struct node * tmp;
   int n = 0;

   if(ennode == NULL)
   {
      printf(" No data found in the List yet.");
   }
   else
   {
      tmp = ennode;
      printf("\n Data in reverse order are :\n");
      while(tmp != NULL)
      {
         printf(" Data in node %d : %d\n", n+1, tmp->num);
         n++;
         tmp = tmp->preptr; // current pointer set with previous node
      }
   }
}
```

Output:

```
 Doubly Linked List : Create and display a doubly linked list :
------------------------------------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 5
 Input data for node 2 : 6
 Input data for node 3 : 7


 Data entered on the list are :
 node 1 : 5
 node 2 : 6
 node 3 : 7

 Data in reverse order are :
 Data in node 1 : 7
 Data in node 2 : 6
 Data in node 3 : 5
_
```

PgmNo.10
Date:

IMPLEMENT STACK USING ARRAY

……..

Aim: Write a program to Implement stack using array

Algorithm:

Step 1 - Include all the header files which are used in the program and define a constant 'SIZE' with specific value.
Step 2 - Declare all the functions used in stack implementation.
Step 3 - Create a one dimensional array with fixed size (int stack[SIZE])
Step 4 - Define a integer variable 'top' and initialize with '-1'. (int top = -1)
Step 5 - In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

push(value) - Inserting value into the stack

Step 1 - Check whether **stack** is **FULL**. (**top == SIZE-1**)
Step 2 - If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
Step 3 - If it is **NOT FULL**, then increment **top** value by one (**top++**) and set stack[top] to value (**stack[top] = value**).

pop() - Delete a value from the Stack

Step 1 - Check whether stack is EMPTY. (top == -1)
Step 2 - If it is EMPTY, then display "Stack is EMPTY!!!" and terminate the function.
Step 3 - If it is NOT EMPTY, then define a variable 'i' and initialize with top.
Display stack[i] value and decrement i value by one (i--).
Step 3 - Repeat above step until i value becomes '0'.

Program:

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{
```

```
   //clrscr();
top=-1;
printf("\n Enter the size of STACK[MAX=100]:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS USING ARRAY");
printf("\n\t ------------------------------ ");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
   {
printf("\n Enter the Choice:");
scanf("%d",&choice);
```

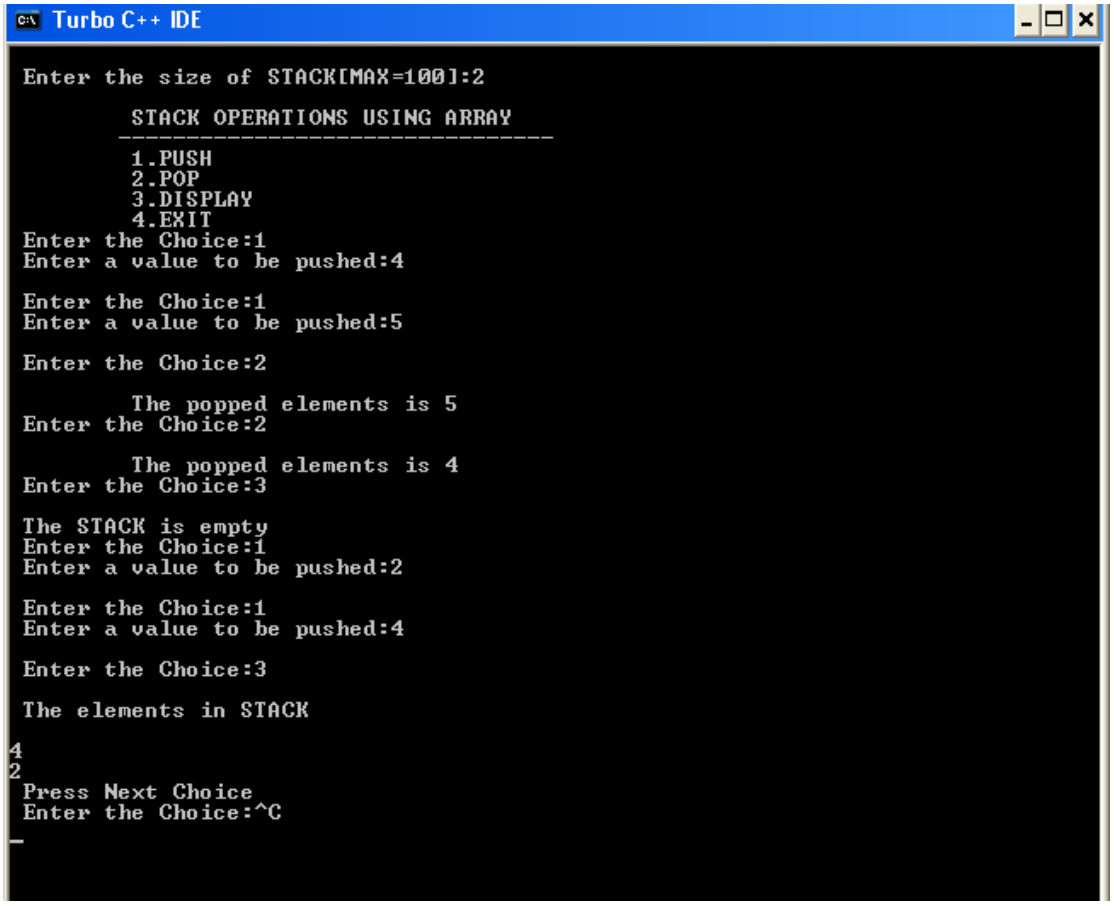
```
switch(choice)
      {
case 1:
         {
push();
break;
         }
case 2:
         {
pop();
break;
         }
case 3:
         {
display();
break;
         }
case 4:
         {
printf("\n\t EXIT POINT ");
break;
         }
default:
         {
printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
         }

      }
```

```
    }
while(choice!=4);
return 0;
}
void push()
{
if(top>=n-1)
    {
printf("\n\tSTACK is over flow");

    }
else
    {
printf(" Enter a value to be pushed:");
scanf("%d",&x);
top++;
stack[top]=x;
    }
}
void pop()
{
if(top<=-1)
    {
printf("\n\t Stack is under flow");
    }
else
    {
printf("\n\t The popped elements is %d",stack[top]);
```

```
top--;
   }
}
void display()
{
   if(top>=0)
   {
        printf("\n The elements in
        STACK \n"); for(i=top; i>=0; i--)
        printf("\n%d",stack[i]);
        printf("\n Press Next
        Choice");
   }
else
   {
printf("\n The STACK is empty");
   }

}
```

Output:

Sample output:



Pgm No.11

Date:

STACK_LINKEDLIST

……..

Aim: Write a program to implement stack using_linkedlist

ALGORITHM:

STEP 1: Start
STEP 2: Declare data,val as integer array and stack *next as struct variable AND *TOP
STEP 3:Initialize *top=NULL
STEP 4:Print main menu PUSH,POP,DISPLAY
STEP 5: Print enter your option
STEP 6: Check option go to step 7 to step 22
STEP 7: Print number to be pushed
STEP 8: Set top=push(top,val) else go to step 9
STEP 9: Set top =pop(top)
STEP 10: Set top=display(top)
STEP 11: Check While(Option!=4)
STEP 12: Check if (top==NULL) else go to step 15
STEP 13: Set ptr ->next=NULL
STEP 14: Set top=ptr
STEP 15:Set ptr->next=top
STEP 16: Set top=ptr
STEP 17:Check if (top—NULL)
STEP 18:Print Stack is empty
STEP 19:Check while(ptr!=NULL)
STEP 20:Read ptr->data
STEP 21:Set ptr =ptr->next
STEP 22:Check if (top==NULL) else goto step 24
STEP 23:Print Stack underflow
STEP 24:Set top=top->next
STEP 25:Print The value being deleted
STEP 26:Set ptr ->data
STEP 27:Stop

Program:

```
#include<stdio.h>

#include<conio.h>

#include<malloc.h>

struct stack

{

 int data;
```

```
 struct stack *next;
};
struct stack *top=NULL;
struct stack *push(struct stack *, int);
struct stack *display(struct stack *);
struct stack *pop(struct stack *);
int main()
{
 int val, option;
 clrscr();
 do
 {
  printf("\nMAIN MENU");
  printf("\n 1. PUSH");
```

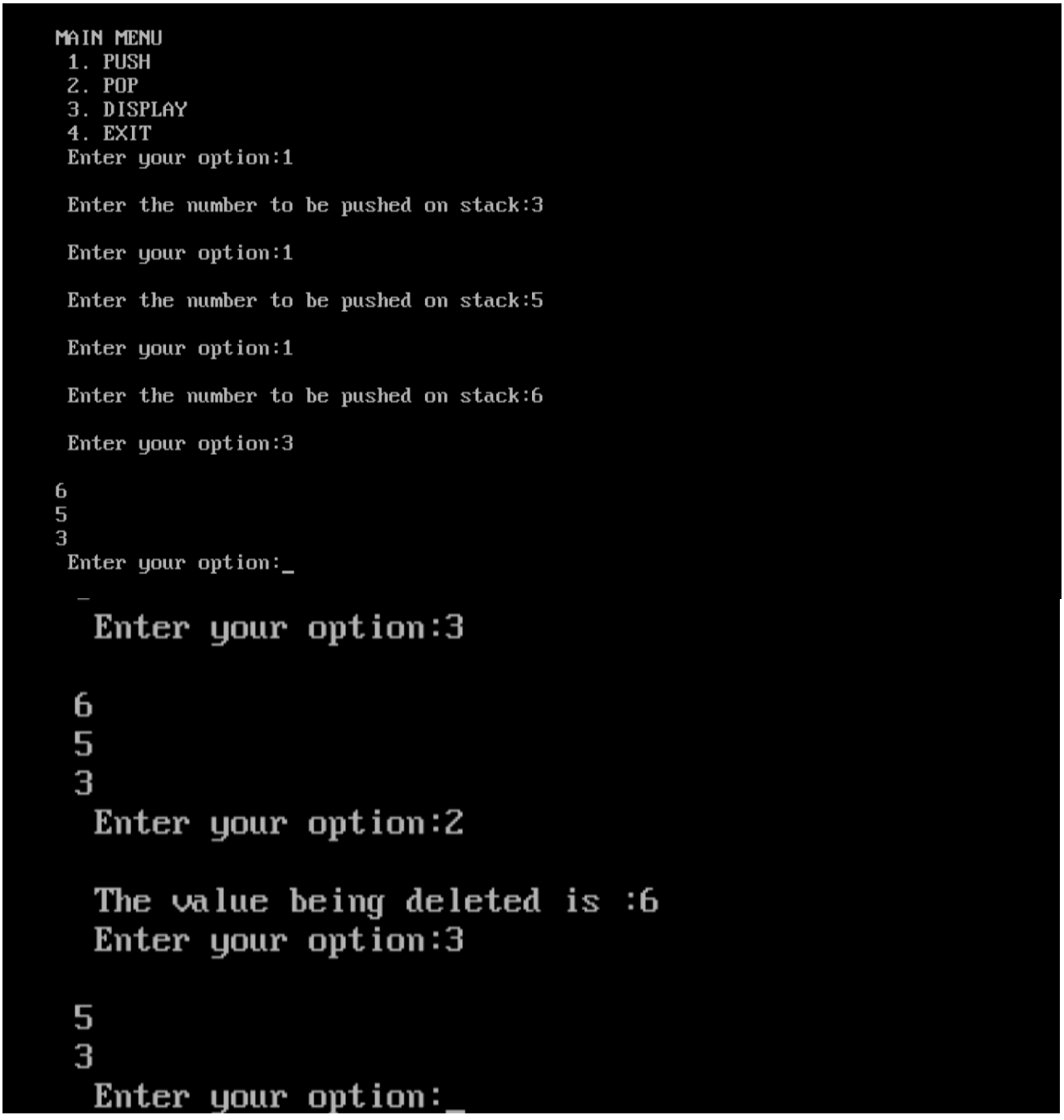
```
 printf("\n 2. POP");
 printf("\n 3. DISPLAY");
 printf("\n 4. EXIT");
 printf("\n Enter your option:");
 scanf("%d", &option);
 switch(option)
 {
  case 1:
     printf("\n Enter the number to be pushed on stack:");
     scanf("%d", &val);
     top=push(top,val);
     break;
  case 2:
     top=pop(top);
     break;
  case 3:
     top=display(top);
     break;
  }
 }while(option !=4);
 getch();
 return 0;
}
struct stack *push(struct stack *top, int val)
{
 struct stack *ptr;
 ptr=(struct stack *)malloc(sizeof(struct stack));
 ptr->data=val;
```

```
 if(top==NULL)
 {
  ptr->next=NULL;
  top=ptr;
 }
 else
 {
  ptr->next=top;
  top=ptr;
 }
 return top;
}

struct stack *display(struct stack *top)
{
 struct stack *ptr;
 ptr=top;
 if(top==NULL)
  printf("\n STACK IS EMPTY");
 else
 {
  while(ptr!=NULL)
  {
   printf("\n%d", ptr->data);
   ptr=ptr->next;
  }
}
 return top;
```

```
}

struct stack *pop(struct stack *top)
{
 struct stack *ptr;
 ptr=top;
 if(top==NULL)
   printf("\n STACK UNDERFLOW");
 else
  {
   top=top->next;
   printf("\n The value being deleted is :%d",ptr->data);
   free(ptr);
  }
 return top;
 }
```

output:

sample output:



PgmNo.12
Date:

POSTFIX EVALUATION

……..

Aim: Write a program to implement evaluation of postfix expression

ALGORITHM:

STEP 1: Start
STEP 2: Declare pop(),push(),postfix() as functions
STEP 3:Declare stack[size],top=1
STEP 4:Declare I,a,b,result and PEval as integer Array
STEP 5: Declare ch as character
STEP 6: Initialize i=0 and check i<size
STEP 7: Print enter a postfix expression
STEP 8: Read postfix
STEP 9: Check postfix[i]!+'/o'and initialize value of i
STEP 10: Set ch=postfix[i]
STEP 11: Check if(is digit) else goto step 13
STEP 12: Set push(ch-'0')
STEP 13: Check (ch=='+',ch='-',ch='*' and ch='/')
STEP 14: Set b=top and a=top
STEP 15:Print stack is empty
STEP 16: Check i<n
STEP 17:Read arr[i]
STEP 18:Stop

Program:

```
#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#define SIZE 40

int pop();

void push(int);

char postfix[SIZE];

int stack[SIZE], top = -1;
```

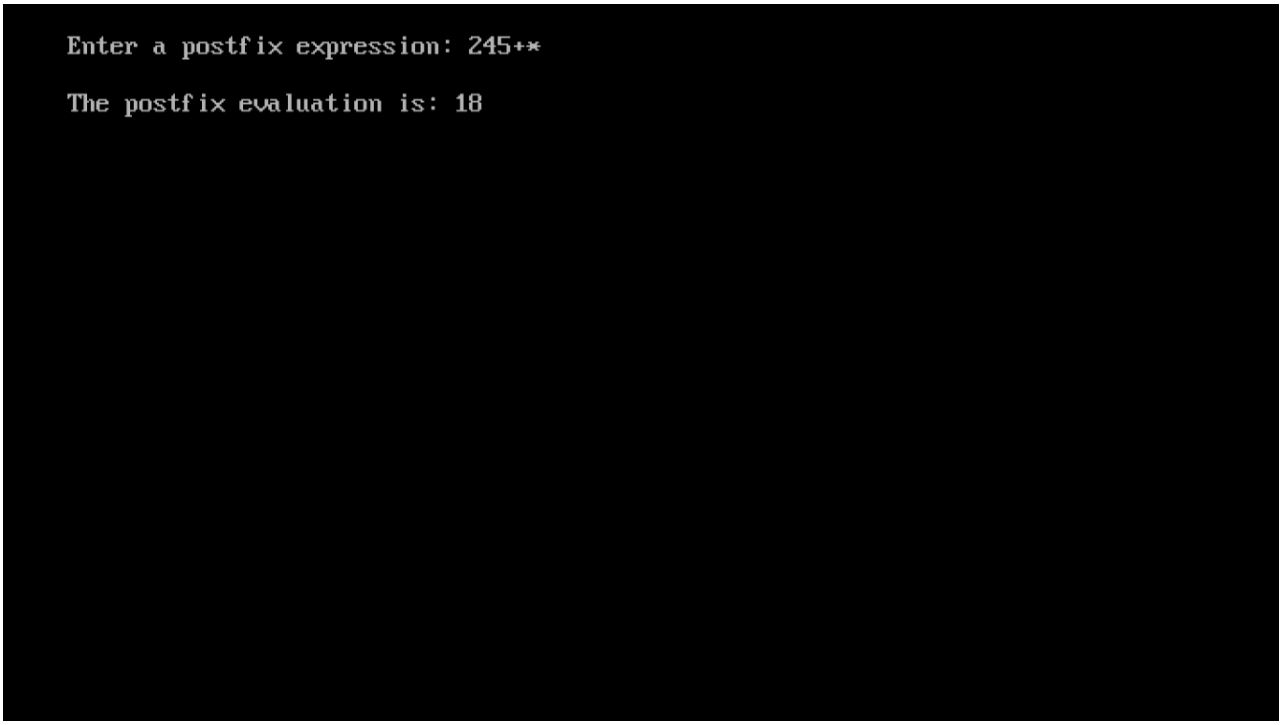
```
void main()
{
    int i, a, b, result, pEval;
    char ch;
    for(i=0; i<SIZE; i++)
    {
        stack[i] = -1;
    }
    printf("\nEnter a postfix expression: ");
    scanf("%s",postfix);
for(i=0; postfix[i] != '\0'; i++)
    {
        ch = postfix[i];

        if(isdigit(ch))
        {
            push(ch-'0');
        }
else if(ch == '+' || ch == '-' || ch == '*' || ch == '/')
{
    b = pop();
    a = pop();
    switch(ch)
    {
        case'+': result = a+b;
```

```
                        break;
        case '-': result = a-b;
                        break;
        case '*': result = a*b;
                        break;
        case '/': result = a/b;
                        break;
        }

        push(result);
                }
        }
        pEval = pop();
printf("\nThe postfix evaluation is:%d\n",pEval);
getch();
}
void push(int n)
    {
        if (top < SIZE -1)
        {
                stack[++top] = n;
        }
        else
        {
                printf("Stack isfull!\n");
```

```
                exit(-1);
        }
    }
    int pop()
    {
        int n;
        if (top > -1)
        {
            n = stack[top];
            stack[top--] = -1;
            return n;
        }
        else
        {
            printf("Stack is empty!\n");
            exit(-1);
            return -1;
        }
    }
    for(i=0;i<n;i++)
        printf(" %d\t", arr[i]);
   getch();
}
```

output:

sample output:



Pgm No.13
Date:

QUEUE USING ARRAY

Aim: Implement Queue using array

ALGORITHM:

STEP 1: Start
STEP 2: Declare queue[n],ch=1,front=0,rear=0,i,j=1,x=n as integer array
STEP 3:Print menu insertion,deletion,display,exit
STEP 4:Check while(ch)
STEP 5: Print enter your choice
STEP 6: Read ch
STEP 7: Check swich(ch) goto step 8 to 16
STEP 8: Check if(rear==x) else goto step 10
STEP 9: Print queue is full
STEP 10: Print enter the number
STEP 11: Read queue[rear++]
STEP 12: Check if (front==rear) else go to step 13
STEP 13: Print queue is empty
STEP 14: Print deleted element is
STEP 15:Increment x by 1
STEP 16: Check if(front==rear)else goto step 17
STEP 17:Print queue is empty
STEP 18:Set i=front
STEP 19:Print queue[i]
STEP 20:Stop

Program:

```
#include<stdio.h>
#define n 5
void main()
{
   int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
   printf("Queue using Array");
   printf("\n1.Insertion \n2.Deletion \n3.Display \n4.Exit");
   while(ch)
   {
      printf("\nEnter the Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
```

```
        case 1:
            if(rear==x)
                printf("\n Queue is Full");
            else
            {
                printf("\n Enter no %d:",j++);
                scanf("%d",&queue[rear++]);
            }
            break;
        case 2:
            if(front==rear)
            {
                printf("\n Queue is empty");
            }
            else
            {
                printf("\n Deleted Element is %d",queue[front++]);
                x++;
            }
            break;
        case 3:
            printf("\nQueue Elements are:\n ");
            if(front==rear)
                printf("\n Queue is Empty");
            else
            {
                for(i=front; i<rear; i++)
                {
                    printf("%d",queue[i]);
                    printf("\n");
                }
                break;
            case 4:
                exit(0);
            default:
                printf("Wrong Choice: please see the options");
            }
        }
    }
getch();
    }
```

output:

sample output

Pgm No.14
Date:

QUEUE USING LIST

……..

Aim: Implement Queue using linked list.

ALGORITHM:

STEP 1: Start
STEP 2: Declare node as struct
STEP 3:Declare data as integer
STEP 4:Declare node *next,*front,*rear as struct variable
STEP 5: Declare menu insert,delete,display
STEP 6: Print menu
STEP 7: Declare choice as integer
STEP 8: Read choice
STEP 9: Check switch(choice) goto step 10 to step18
STEP 10: Check cases for insert,delete and display
STEP 11: Print enter a valid choice
STEP 12: Check if(ptr==NULL) else goto step 14
STEP 13: Print Overflow
STEP 14: Read item
STEP 15:Set ptr->data=item
STEP 16: Check if(front==NULL)else goto step 18
STEP 17:Set front=ptr
Rear=ptr
front->next=NULL
STEP 18:Set rear->next=ptr
rear->ptr
rear->next=NULL
STEP 19:Check while(ptr!=NULL)
STEP 20: Print ptr->data
STEP 21:Stop

PROGRAM:

```
void main()
{

#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
```

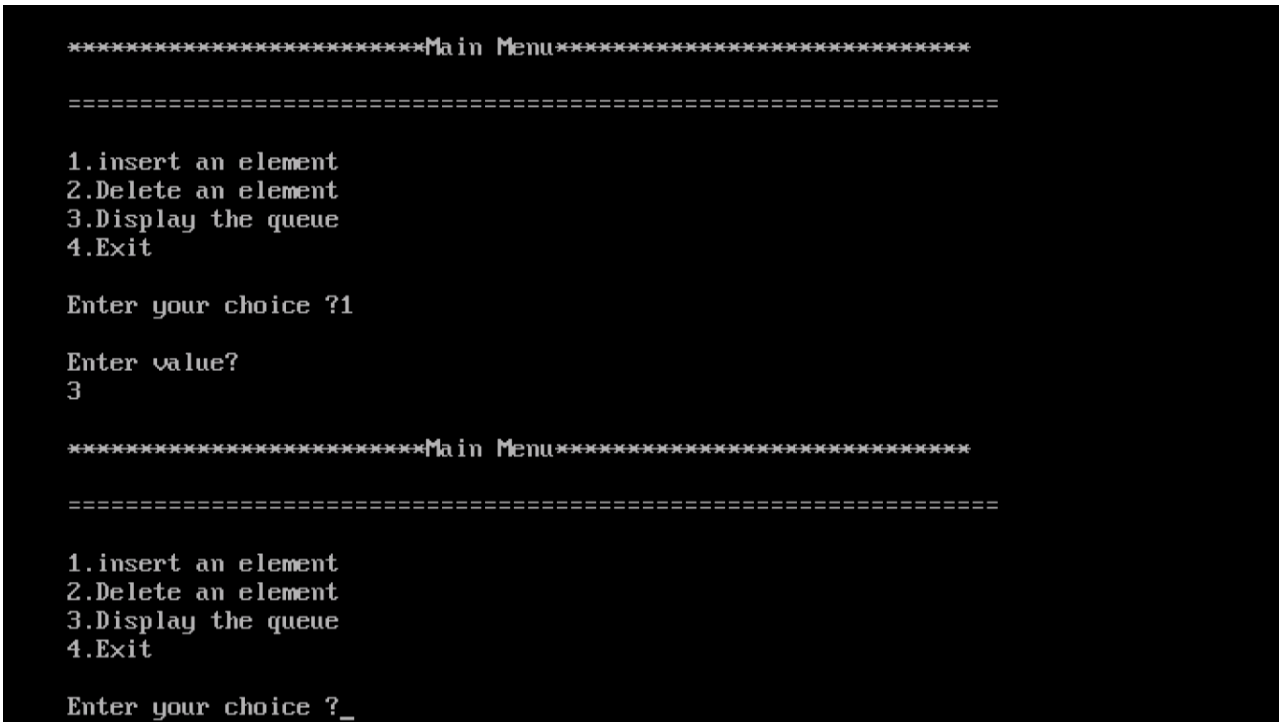
```
    struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
    int choice;
    while(choice != 4)
    {
        printf("\n*************************Main
Menu*****************************\n");

printf("\n==========================================================
========\n");
        printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
        printf("\nEnter your choice ?");
        scanf("%d",& choice);
        switch(choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(0);
            break;
            default:
            printf("\nEnter valid choice??\n");
        }
    }
}
void insert()
{
    struct node *ptr;
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
```

```
            return;
        }
        else
        {
            printf("\nEnter value?\n");
            scanf("%d",&item);
            ptr -> data = item;
            if(front == NULL)
            {
                front = ptr;
                rear = ptr;
                front -> next = NULL;
                rear -> next = NULL;
            }
            else
            {
                rear -> next = ptr;
                rear = ptr;
                rear->next = NULL;
            }
        }
    }
    void delete ()
    {
        struct node *ptr;
        if(front == NULL)
        {
            printf("\nUNDERFLOW\n");
            return;
        }
        else
        {
            ptr = front;
            front = front -> next;
            free(ptr);
        }
    }
    void display()
    {
        struct node *ptr;
        ptr = front;
        if(front == NULL)
        {
            printf("\nEmpty queue\n");
        }
        else
        {   printf("\nprinting values .....\n");
            while(ptr != NULL)
```

```
        {
            printf("\n%d\n",ptr -> data);
            ptr = ptr -> next;
        }
    }
}
```

Output

Sample output



```
=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....

3

6

**************************Main Menu*********************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?_
```

Pgm No.15

Date:

BINARY SEARCH TREE

……..

Aim: Search an element in a binary search tree

ALGORITHM:

STEP 1: Start
STEP 2: Declare node as struct
STEP 3:Declare info ,LOC,PAR,info as integer
STEP 4:Declare *left and right as struct
STEP 5: Declatre type def BST
STEP 6: Check if(root==NULL)
STEP 7: Set LOC=NULL and PAR=NULL
STEP 8: Check if(item==root->info)
STEP 9: Set Loc=root and PAR=NULL
STEP 10: Check if(item<root->info) else goto step 12
STEP 11: Set save=root
Ptr=root->left
STEP 12: Set save =root
ptr=root->right
STEP 13: Declare x,c=1,z,element as an integer and ch as character
STEP 14: Check while (ch=='y')
STEP 15:Read x
STEP 16: Set root=insert(root,x)
STEP 17:Check cases for switch else goto step 21
STEP 18:Printf Enter the item
STEP 19: Read &z
STEP 20: Set root = insert(root,z)
STEP 21:Print enter a valid choice
STEP 22:Stop

PROGRAM:

```
void main()
{
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int info;
        struct node*left;
        struct node*right;
};
typedef struct node BST;
BST *LOC, *PAR;
void search(BST *root, int item)
{
   BST *save,*ptr;
   if (root == NULL)
   {
      LOC = NULL;
      PAR=NULL;
   }
   if (item == root -> info)
   {
   LOC = root;
   PAR = NULL;
   return;
   }
   if (item < root->info)
   {
      save = root;
      ptr = root->left;
   }
   else
   {
      save = root;
      ptr = root -> right;
   }
   while( ptr != NULL)
   {
      if (ptr -> info == item)
      {
         LOC = ptr;
         PAR = save;
         return;
      }
      if(item < ptr->info)
      {
```

```
            save = ptr;
            ptr = ptr->left;
        }
        else
        {
            save = ptr;
            ptr = ptr->right;
        }
    }
    LOC = NULL;
    PAR = save;
    return;
}

struct node*insert(struct node*r, int x)
{
         if (r == NULL)
         {
          r = (struct node*)malloc(sizeof(struct node));
          r->info = x;
          r->left = r->right = NULL;
          return r;
         }
         else if (x < r->info)
          r->left = insert(r->left, x);
         else if (x > r->info)
          r->right = insert(r->right, x);
         return r;
}
void main()
{
    struct node* root = NULL;
    int x, c = 1, z;
    int element;
    char ch;
    printf("\nEnter an element: ");
    scanf("%d", &x);
    root = insert(root, x);
    printf("\nDo you want to enter another element :y or n");
    scanf(" %c",&ch);
    while (ch == 'y')
    {
        printf("\nEnter an element:");
        scanf("%d", &x);
        root = insert(root,x);
        printf("\nPress y or n to insert another element: y or n: ");
        scanf(" %c", &ch);
    }
```

```
    while(1)
    {
       printf("\n1 Insert an element ");
       printf("\n2 Search for an element ");
       printf("\n3 Exit ");
       printf("\nEnter your choice: ");
       scanf("%d", &c);
       switch(c)
       {
          case 1:
             printf("\nEnter the item:");
             scanf("%d", &z);
             root = insert(root,z);
             break;

          case 2:
             printf("\nEnter element to be searched:  ");
             scanf("%d", &element);
             search(root, element);
             if(LOC != NULL)
                printf("\n%d Found in Binary Search Tree !!\n",element);
             else
                printf("\nIt is not present in Binary Search Tree\n");
             break;
          case 3:
             printf("\nExiting...");
                  return;
          default:
             printf("Enter a valid choice: ");

       }
    }
    getch();
}
```

Output:

Sample output:

```
Enter an element: 2

Do you want to enter another element :y or ny

Enter an element:4

Press y or n to insert another element: y or n: n

1 Insert an element
2 Search for an element
3 Exit
Enter your choice: 2

Enter element to be searched:  7

It is not present in Binary Search Tree

1 Insert an element
2 Search for an element
3 Exit
Enter your choice: _
```

```
Press y or n to insert another element: y or n: n

1 Insert an element
2 Search for an element
3 Exit
Enter your choice: 2

Enter element to be searched:  7

It is not present in Binary Search Tree

1 Insert an element
2 Search for an element
3 Exit
Enter your choice: 2

Enter element to be searched:  4

4 Found in Binary Search Tree !!

1 Insert an element
2 Search for an element
3 Exit
Enter your choice:
```

PgmNo.16
Date:

IMPLEMENTATION EXCHANGE SORT

……..

Aim: Implement exchange sort

ALGORITHM

STEP 1: Start
STEP 2: Declare arr[] as integer array
STEP 3:Declare i,n ,temp as integer variable.
STEP4:Read the size of f array in n
STEP5: Read the elements of array in arr [i]
STEP6:Repeat step 7 to 9for i=0 to n-1
STEP7: Repeat step 8 to 9 for j=i+1 to n
STEP8:check arr[i]>arr[j]
STEP9::set temp= arr [i], arr [i]=arr[j] and arr[j]=temp
STEP10::Print the sorted array
STEP11:Stop

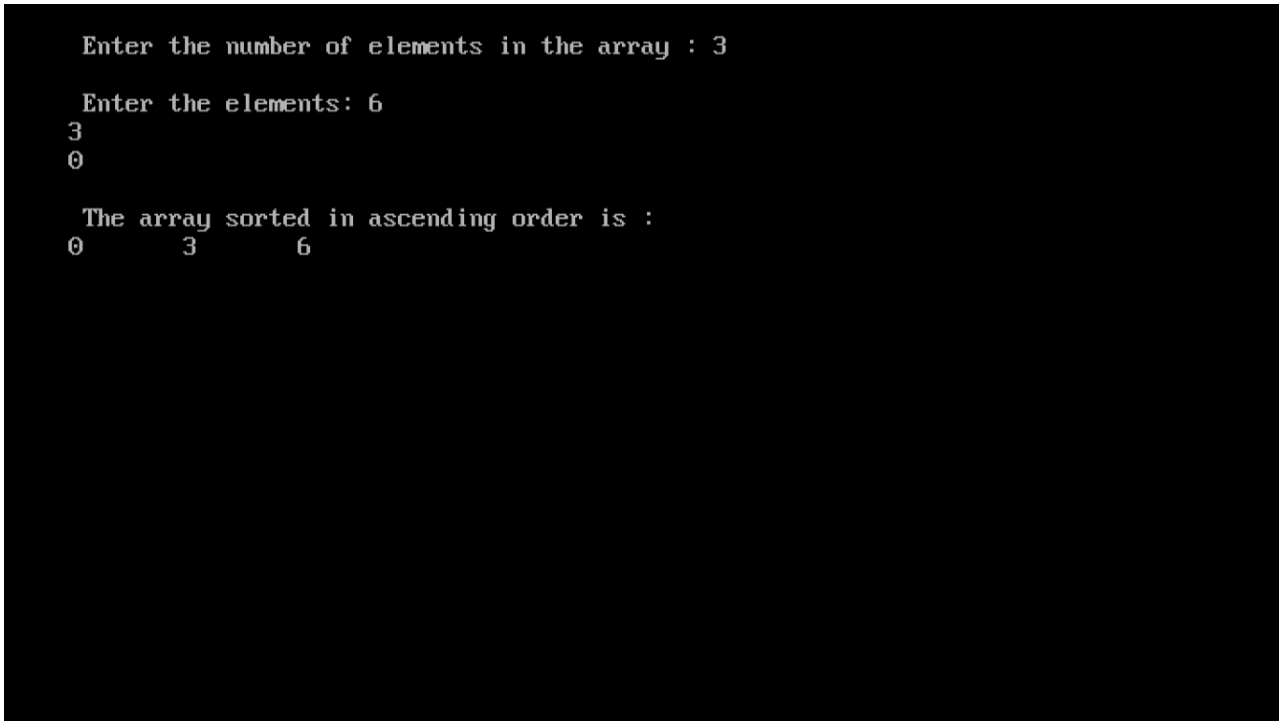
Program:

```
#include <stdio.h>
 #include <conio.h>
 Void  main()
 {
        int i, n, temp=0, j, arr[10];
        clrscr();
        printf("\n Enter the number of elements in the array : ");
        scanf("%d", &n);
      printf("\n Enter the elements:
 ");
        for(i=0;i<n;i++)
        {
```

```
            scanf("%d", &arr[i]);
        }
        for(i=0;i<n-1;i++)
        {
            for(j=i+1;j<n;j++)
            {
                if(arr[i] > arr[j])
                {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                }
            }
        }
printf("\n The array sorted in ascending order is
:\n");
        for(i=0;i<n;i++)
        {
            printf("%d\t", arr[i]);
        }

getch();
}
```

Output

Sample output



PgmNo.17
Date:

IMPLEMENTATION SELECTIONSORT

……..

Aim: Write a program to implementation selectionsort

Algorithm:

Algorithm for main()

STEP 1: Start
STEP 2: Declare arr[] as integer array
STEP 3:Declare i,n as integer variable.
STEP4:Read the size of f array in n
STEP5: Read the elements of array in arr [i]
STEP6:call the function insertion_sort(arr[],n)
STEP7Print the sorted array
STEP 8: Stop

Algorithm for selection_sort(arr[],n)

STEP 1: Start
STEP 2 : Declare k,pos,temp as integer variable.
STEP 3: Repeat step 4 and 5 for i=0 to n
STEP 4 : call smallest(arr, k, n) and set to pos
STEP 5 : set temp= arr [k], arr [k]=arr[pos] and arr[pos]=temp
STEP 6 :Stop

Algorithm for smallest(arr[],k,n)

STEP 1: Start
STEP 2 :set i as integer, pos=k, small =arr[k]
STEP 3: Repeat step 4 for i=k+1 to n
STEP 4 : check small>arr[k] then set small=arr[i] and pos=i
STEP 5 : return the values
STEP 6 :Stop

Program:

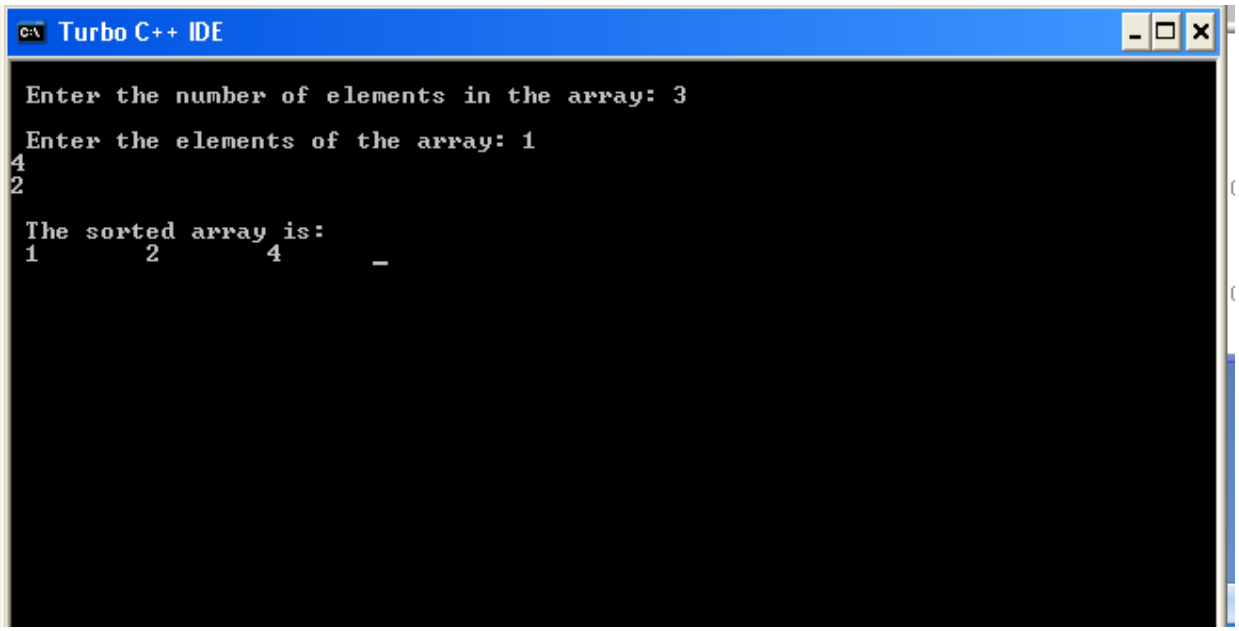
```
#include <stdio.h>

#include <conio.h>

int smallest(int arr[], int k, int n);

void selection_sort(int arr[], int n);
```

```
void main()
{
        int arr[10], i, n;
        clrscr();
        printf("\n Enter the number of elements in the array: ");
        scanf("%d", &n);
        printf("\n Enter the elements of the array: ");
        for(i=0;i<n;i++)
        {
                scanf("%d", &arr[i]);
        }
        selection_sort(arr, n);
```

```
        printf("\n The sorted array is: \n");
        for(i=0;i<n;i++)
                printf(" %d\t", arr[i]);
getch();
}
int smallest(int arr[], int k, int n)
{
        int pos = k, small=arr[k], i;
        for(i=k+1;i<n;i++)
        {
                if(arr[i]< small)
                {
                        small = arr[i];
                        pos = i;
                }
        }
return pos;
}
void selection_sort(int arr[],int n)
{
        int k, pos, temp;
        for(k=0;k<n;k++)
        {
                pos = smallest(arr, k, n);
                temp = arr[k];
                arr[k]=arr[pos];
                arr[pos]=temp;
        }
```

Output

Sample Output



Pgm No.18
Date:

IMPLEMENTATION INSERTION SORT

……..

Aim: Write a program to implementation insertionsort

Algorithm:

Algorithm for main()
STEP 1: Start
STEP 2: Declare arr[] as integer array
STEP 3:Declare i,n as integer variable.
STEP4:Read the size of f array in n
STEP5: Read the elements of array in arr [i]
STEP6:call the function insertion_sort(arr[],n)
STEP7Print the sorted array
STEP 8: Stop
Algorithm for insertion_sort(arr[],n)
STEP 1: Start
STEP 2 Declare I,j,temp as integer variables.
STEP 3: Repeat step 4 to step 8 for i=1 to n
STEP4:set temp=arr[i] and j=i-1
STEP5: Repeat step 6 to 7 while temp<arr[j]
STEP 6:set arr[j+1]=a[j]
STEP 7: set j=j-1
STEP 8::set arr[j+1]=temp
STEP 9:return the values
STEP 10 :stop

Program:

```
#include  <stdio.h>

#include <conio.h>

#define size 5

void insertion_sort(int arr[], int n);

void main()

{

int arr[size], i, n;
```

```
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter the elements of the array: ");
for(i=0;i<n;i++)
{
scanf("%d", &arr[i]);
}
insertion_sort(arr, n);
printf("\n The sorted array is: \n");
for(i=0;i<n;i++)
```

```
printf(" %d\t", arr[i]);
getch();
}
void insertion_sort(int arr[], int n)
{
int i, j, temp;
for(i=1;i<n;i++)
{
temp = arr[i];
j = i-1;
while((temp < arr[j]) && (j>=0))
{
arr[j+1] = arr[j];
j--;
}
arr[j+1] = temp;
}
}
```

Output:

Sample output: