Ex. No: 1

STACK USING ARRAY

Date:

AIM

To create a C program to perform the operations of a stack using array.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method for performing the operations of a stack using array.

STEP 4: Create a one dimensional array with fixed size (**int stack[SIZE]**)

STEP 5: Define a integer variable **'top'** and initialize with **'-1'**. (**int top = -1**)

STEP 6: In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack

STEP 7: Save and compile the program

STEP 8: Run the program & Display the result.

STEP 9: Stop the program.

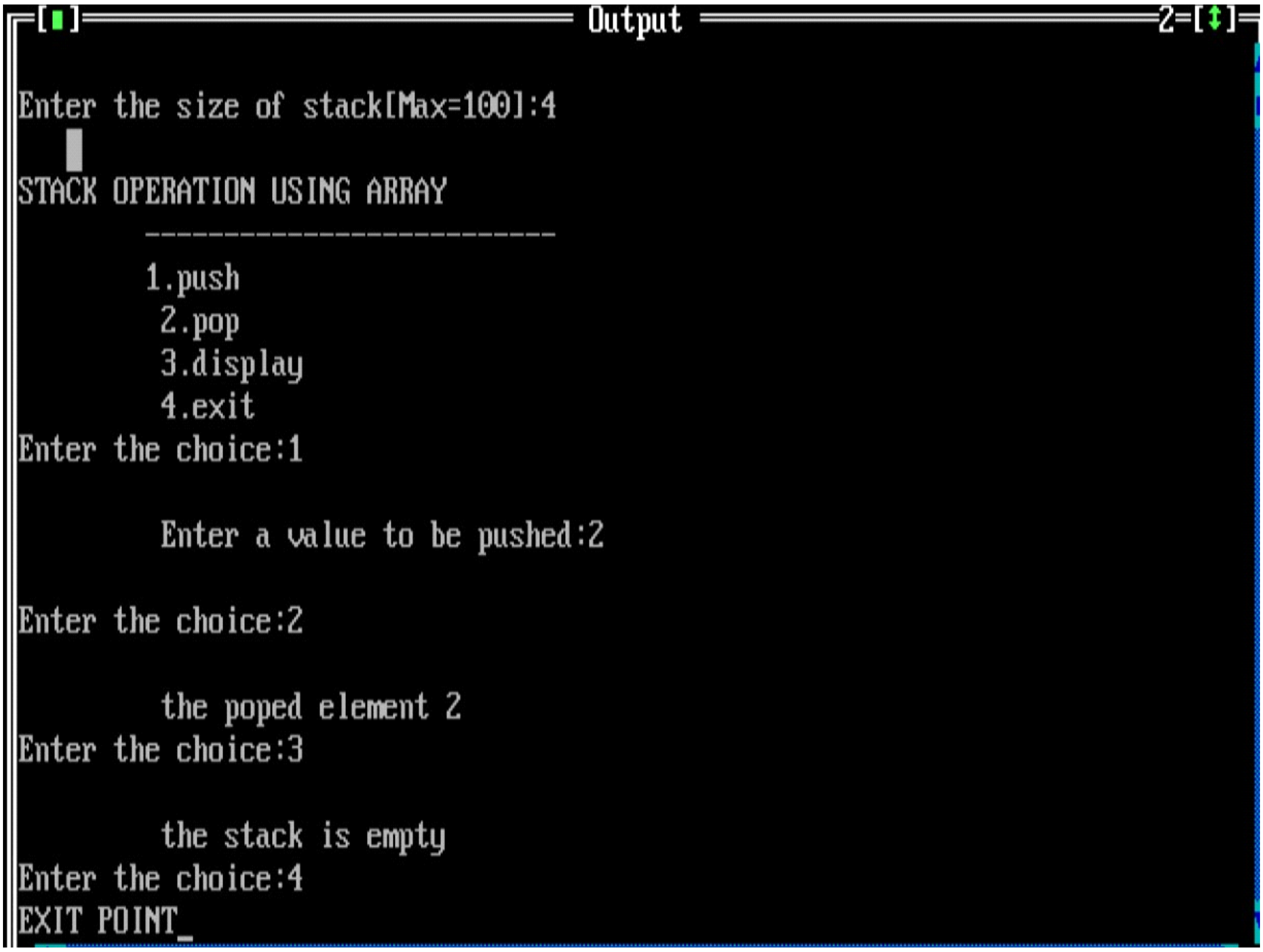
PROGRAM

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
top=-1;
printf("\nEnter the size of stack[Max=100]:");
scanf("%d",&n);
printf("\nSTACK OPERATION USING ARRAY");
printf("\n\t--------------------------");
printf("\n\t1.push\n\t 2.pop\n\t 3.display\n\t 4.exit");
do
{
printf("\nEnter the choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
```

```
display();
break;
}
case 4:
{
printf("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\t{ stack is overflow");
}
else
{
printf("\n\t 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 poped element %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\t the stack is empty");
}
}
```

OUTPUT



```
Enter the size of stack[Max=100]:4

STACK OPERATION USING ARRAY
        --------------------------
        1.push
         2.pop
         3.display
         4.exit
Enter the choice:1

         Enter a value to be pushed:2

Enter the choice:2

         the poped element 2
Enter the choice:3

         the stack is empty
Enter the choice:4
EXIT POINT_
```

RESULT

Thus the program has been executed successfully.

Ex. No: 2

EVALUATING THE POSTFIX EXPRESSION

Date:

AIM

To create a C program to evaluate the postfix expression

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Create a stack to store operands.

STEP 4: Scan the given expression from left to right.

STEP 5: a) If the scanned character is an operand, push it into the stack.

b) If the scanned character is an operator, POP 2 operands from stack and perform operation and PUSH the result back to the stack.

STEP 6: Repeat step 3 till all the characters are scanned.

STEP 7: When the expression is ended, the number in the stack is the final result.

STEP 8: Save and compile the program

STEP 9: Run the program & Display the result.

STEP 10: Stop the program.

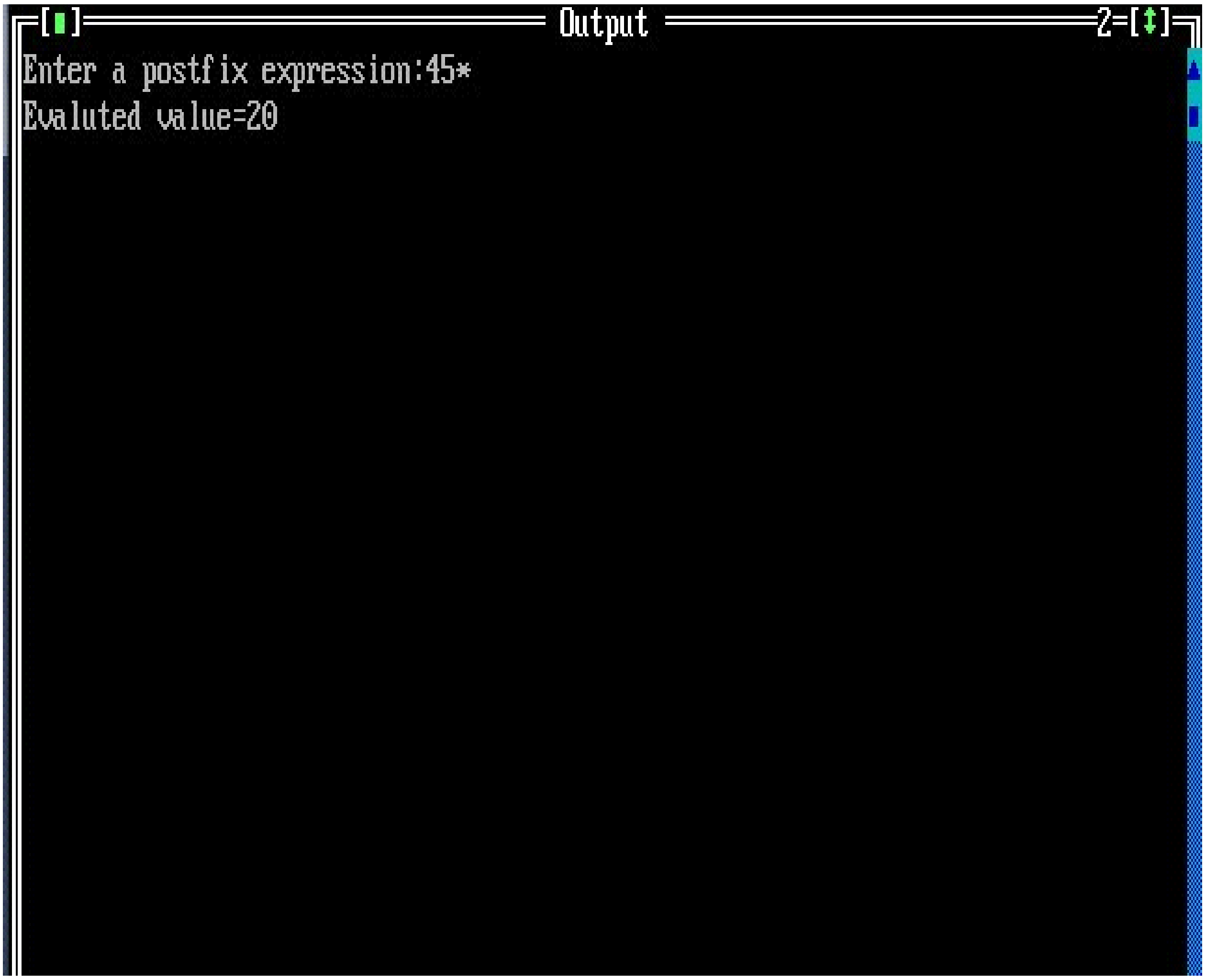
PROGRAM

```
#include<stdio.h>
int top=-1,stack[100];
int main()
{
char a[50],ch;
int i,op1,op2,res,x,v;
void push(int);
int pop();
int eval(char,int,int);
clrscr();
printf("Enter a postfix expression:");
gets(a);
for(i=0;a[i]!='\0';i++)
{
ch=a[i];
if(isalpha(ch))
{
printf("enter value for%c",ch);
scanf("%d",&v);
push(v);
}
if(isdigit(ch))
push(ch-'0');
else if(ch=='+'||ch=='-'||ch=='*'||ch=='/')
{
op2=pop();
op1=pop();
res=eval(ch,op1,op2);
push(res);
}
}
```

```
x=pop();
printf("Evaluted value=%d",x);
getch();
return 0;
}
void push(int n)
{
if(top>=100-1)
{
printf("stack overflow");
return;
}
else
{
top++;
stack[top]=n;
}
}
int pop()
{
int res;
if(top<0)
{
printf("stack underflow");
}
else
{
res=stack[top];
top--;
return res;
}
return 0;
```

```
}
int eval(char ch,int op1,int op2)
{
switch(ch)
{
case '+':return(op1+op2);
case'-':return(op1-op2);
case'*':return(op1*op2);
case'/':return(op1/op2);
}
return 0;
}
```

OUTPUT



```
Enter a postfix expression:45*
Evaluted value=20
```

RESULT

Thus the program has been executed successfully.

Ex. No: 3

QUEUE USING ARRAY

Date:

AIM

To create a C program to perform the Queue operations using array.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Declare all the user defined functions which are used in queue implementation.

STEP 4: Create a one dimensional array with above defined SIZE (**int queue[SIZE]**)

STEP 5: Define two integer variables **'front'** and '**rear**' and initialize both with **'-1'**.

(**int front = -1, rear = -1**)

STEP 6: Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

STEP 7: Save and compile the program

STEP 8: Run the program & Display the result.

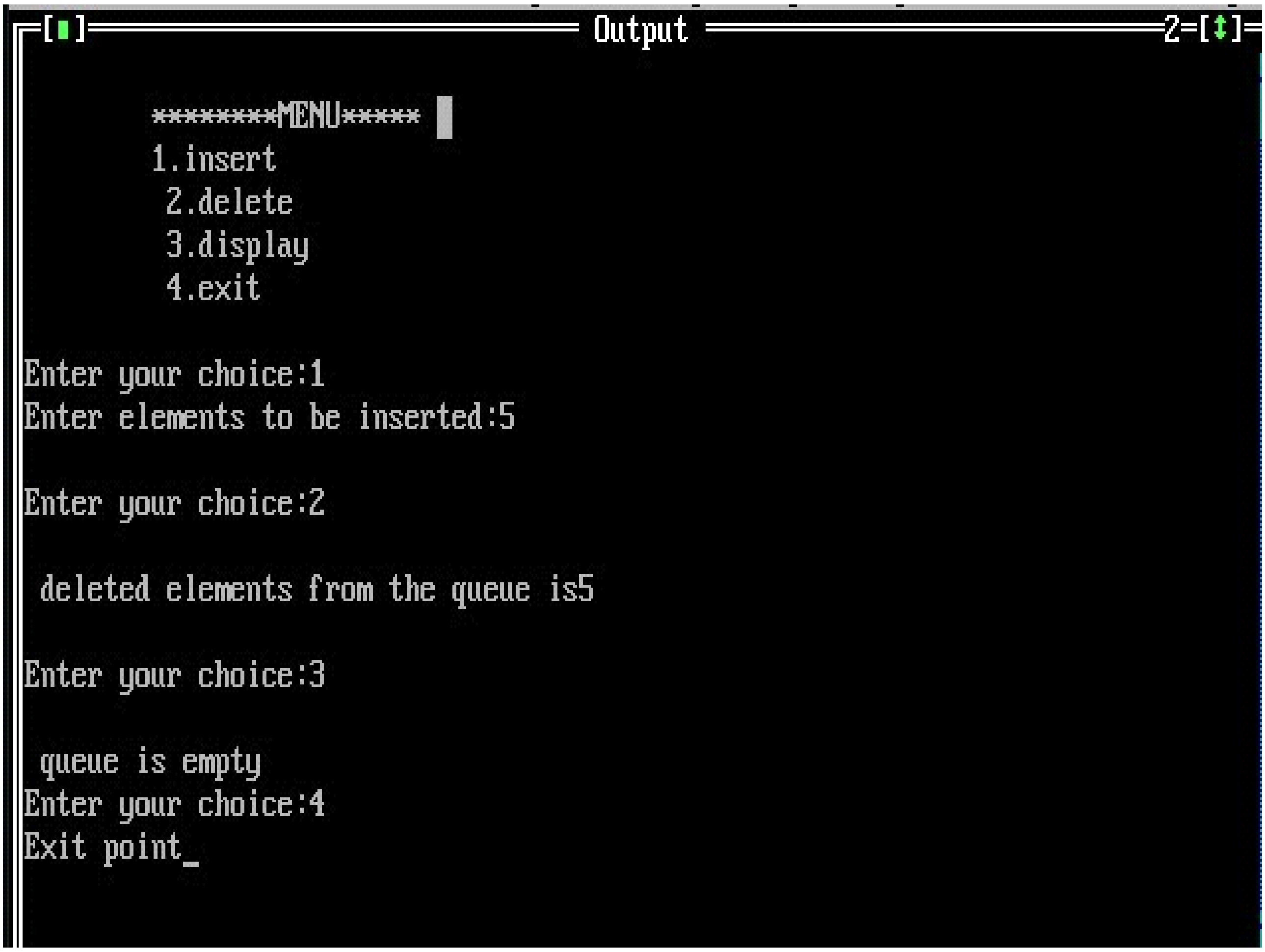
STEP 9: Stop the program.

PROGRAM

```
#include<stdlib.h>
#include<conio.h>
#define MAX 10
void insertion();
void  deletion();
void  display();
int i,choice,rear,front,queue[MAX],items;
int main()
{
front=0;
rear=-1;
printf("\n\t********MENU*****\t\n\t1.insert\n\t 2.delete\n\t 3.display\n\t
4.exit\n\t");
do
{
printf("\nEnter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
insertion();
break;
case 2:
deletion();
break;
case 3:
display();
break;
case 4:
printf("Exit point");
break;
default:
printf("\n\tWRONG CHOICE");
}
}
while(choice!=4);
return 0;
}
void insertion()
{
if(rear>=MAX-1)
printf("\n queue overflow\n");
else
{
printf("Enter elements to be inserted:");
scanf("%d",&items);
rear++;
queue[rear]=items;
}
}
void deletion()
{if(front>rear)
printf("\nqueue underflow");
else
{
items=queue[front];
for(i=0;i<=rear;i++)
queue[i]=queue[i+1];
rear--;
printf("\n deleted elements from the queue is%d\n",items);
}
}
void display()
```

```
{
int i;
if(rear==-1)
printf("\n queue is empty");
else
{
printf("queue content are:\n");
for(i=front;i<=rear;i++)
printf("%d at %d \n",queue[i],i);
}
}
```

OUTPUT



```
        *********MENU******
        1.insert
         2.delete
         3.display
         4.exit

Enter your choice:1
Enter elements to be inserted:5

Enter your choice:2

 deleted elements from the queue is5

Enter your choice:3

 queue is empty
Enter your choice:4
Exit point_
```

RESULT

Thus the program has been executed successfully.

Ex. No: 4

LINKED LIST IMPLEMENTATION OF STACK

Date:

AIM

To create a C program to implement the linked list implementation of stack.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions to create the linked list.

STEP 3: Get the input values to implement the linked list implementation of stack.

STEP 4: Implement the function to insert and delete the elements in a defined position.

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program

STEP 8: Stop the program.

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
   int choice=0;
   printf("\n*********Stack operations using linked list*********\n");
   printf("\n----------------------------------------------\n");
   while(choice != 4)
   {
      printf("\n\nChose one from the below options...\n");
      printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
      printf("\n Enter your choice \n");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
         case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("Exiting....");
                break;
            }
            default:
            {
                printf("Please Enter valid choice ");
            }
    };
}
}
void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
        {
            ptr->val = val;
            ptr -> next = NULL;
            head=ptr;
        }
        else
        {
            ptr->val = val;
            ptr->next = head;
            head=ptr;

        }
        printf("Item pushed");

    }
}

void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");

    }
}
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

OUTPUT

RESULT

Thus the program has been executed successfully.

Ex. No: 5

LINKED LIST IMPLEMENTATION OF QUEUE

Date:

AIM

To create a C program to implement the linked list implementation of queue.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions to create the linked list.

STEP 3: Get the input values to implement the linked list implementation of queue

STEP 4: Implement the function to append & deleting an element the elements in a defined position.

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program & Display the result.

STEP 8: Stop the program.

PROGRAM

```
#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("\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

```
***********Main Menu**********


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

Enter your choice ?1

Enter value?
```

```
123

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


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

Enter your choice ?1

Enter value?
90

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


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

Enter your choice ?3

printing values .....

123

90

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


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

Enter your choice ?2

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

1.insert an element
2.Delete an element
3.Display the queue
```

```
4.Exit

Enter your choice ?3

printing values .....

90

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

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

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

Enter your choice ?4
```

RESULT

Thus the program has been executed successfully.

Ex. No: 6

SEQUENTIAL SEARCH

Date:

AIM

To create a C program to implement the sequential search.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Implement the function to process the sequential search

STEP 4: Implement the function to get the input and search key

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program & Display the result.

STEP 8: Stop the program.

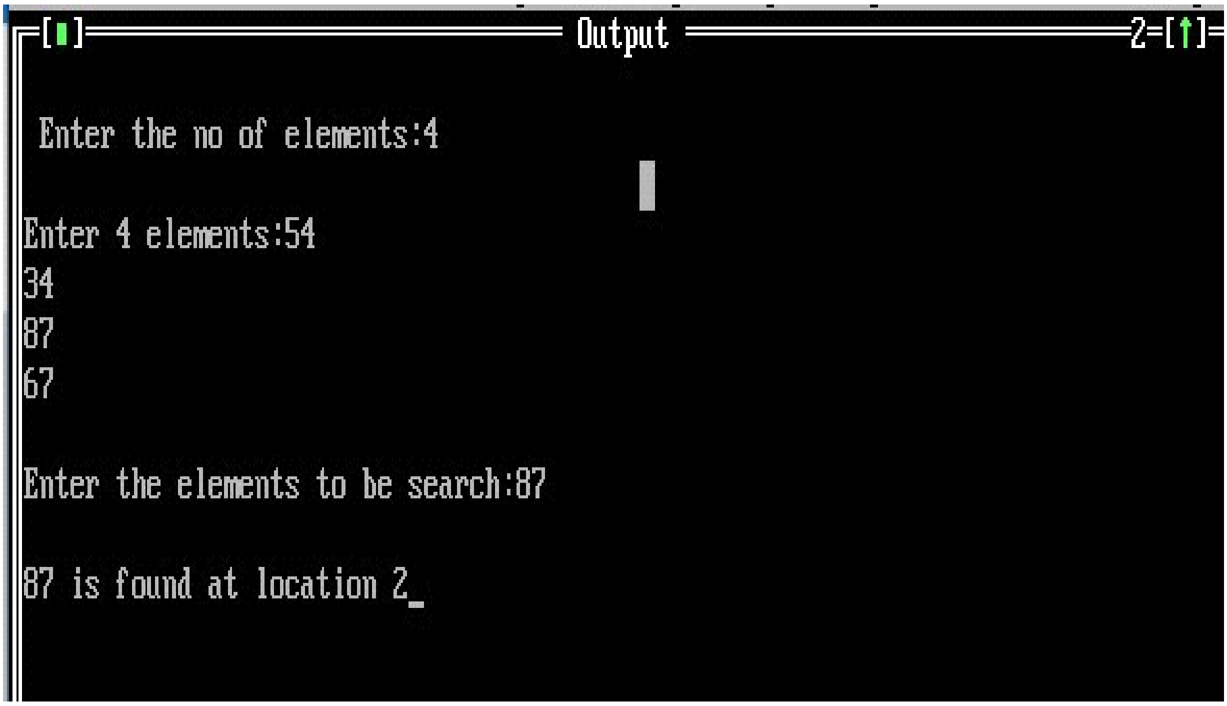
PROGRAM

```
#include<stdio.h>
#include<conio.h>
int sequentialsearch(int[],int,int);

void main()
{
int x[20],i,n,p,key;
clrscr();
printf("\n Enter the no of elements:");
scanf("%d",&n);
printf("\nEnter %d elements:",n);
for(i=0;i<n;i++)
scanf("%d",&x[i]);
printf("\nEnter the elements to be search:");
scanf("%d",&key);
p=sequentialsearch(x,n,key);
if(p==-1)
printf("\n the search is unsuceessful\n");
else
printf("\n%d is found at location %d",key ,p);
getch();
}
int sequentialsearch(int a[],int n,int k)
{
int i;
for(i=0;i<n;i++)
{
if(k==a[i])

return(i);
}
return(-1);
}
```

OUTPUT



```
 Enter the no of elements:4

Enter 4 elements:54
34
87
67

Enter the elements to be search:87

87 is found at location 2_
```

RESULT

Thus the program has been executed successfully.

Ex. No: 7

BINARY SEARCH

Date:

AIM

To create a C program to implement the binary search.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Implement the function to process the binary search

STEP 4: Implement the function to get the input and search key

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program & Display the result.

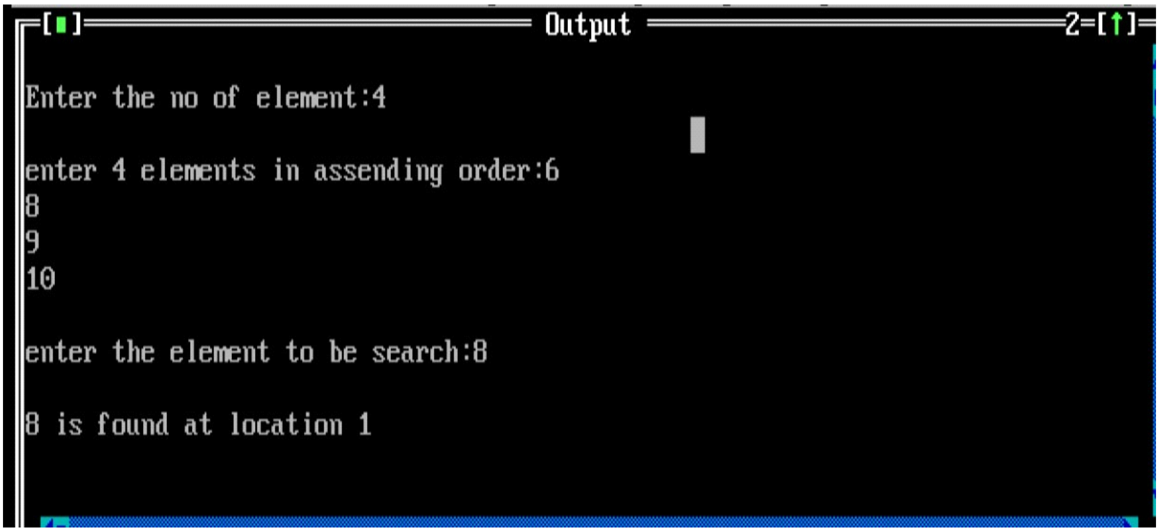
STEP 8: Stop the program.

PROGRAM

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

```
#include<conio.h>
int binarysearch(int[],int,int);
void main()
{
int x[20],i,n,p,key;
clrscr();
printf("\nEnter the no of element:");
scanf("%d",&n);
printf("\nenter %d elements in assending order:",n);
for(i=0;i<n;i++)
scanf("%d",&x[i]);
printf("\nenter the element to be search:");
scanf("%d",&key);
p=binarysearch(x,n,key);
if(p==-1)
printf("\n the search is unsuccessful:\n");
else
printf("\n%d is found at location %d",key,p);
getch();
}
int binarysearch(int a[],int n,int k)
{
int lo,hi,mid;
lo=0;
hi=n-1;
while(lo<=hi)
{
mid=(lo+hi)/2;
if(k==a[mid])
return(mid);
if(k<a[mid])
hi=mid-1;
else
lo=mid+1;
}
return(-1);
}
```

OUTPUT



```
Enter the no of element:4

enter 4 elements in assending order:6
8
9
10

enter the element to be search:8

8 is found at location 1
```

RESULT

Thus the program has been executed successfully.

Ex. No: 8

INORDER TREE TRAVERSAL

Date:

AIM

To create a C program for Inorder traversal of the binary search tree

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Implement the function to process the inorder tree traversal.

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program

STEP 8: Stop the program.

PROGRAM

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

struct node
{
    int key;
    struct node *left;
    struct node *right;
};

//return a new node with the given value
struct node *getNode(int val)
{
    struct node *newNode;

    newNode = malloc(sizeof(struct node));

    newNode->key   = val;
    newNode->left  = NULL;
    newNode->right = NULL;

    return newNode;
}

//inserts nodes in the binary search tree
struct node *insertNode(struct node *root, int val)
{
     if(root == NULL)
         return getNode(val);

     if(root->key < val)
         root->right = insertNode(root->right,val);

     if(root->key > val)
         root->left = insertNode(root->left,val);

     return root;
}

//inorder traversal of the binary search tree
void inorder(struct node *root)
```

```
  {
     if(root == NULL)
    return;

     //traverse the left subtree
     inorder(root->left);

     //visit the root
     printf("%d ",root->key);

     //traverse the right subtree
     inorder(root->right);
  }

  int main()
  {
    struct node *root = NULL;


     int data;
     char ch;
        /*  Do while loop to display various options to select from to decide the input
 */
        do
        {
           printf("\nSelect one of the operations::");
           printf("\n1. To insert a new node in the Binary Tree");
           printf("\n2. To display the nodes of the Binary Tree(via Inorder Traversal).\
n");

           int choice;
           scanf("%d",&choice);
           switch (choice)
           {
           case 1 :
              printf("\nEnter the value to be inserted\n");
              scanf("%d",&data);
              root = insertNode(root,data);
              break;
           case 2 :
              printf("\nInorder Traversal of the Binary Tree::\n");
```

```
                inorder(root);
                break;
            default :
                printf("Wrong Entry\n");
                break;
            }

            printf("\nDo you want to continue (Type y or n)\n");
            scanf(" %c",&ch);
        } while (ch == 'Y'|| ch == 'y');

  return 0;
}
```

OUTPUT

```
Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree (via Inorder Traversal).
1

Enter the value to be inserted
12

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Inorder Traversal).
1

Enter the value to be inserted
98

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Inorder Traversal).
1

Enter the value to be inserted
23

Do you want to continue (Type y or n)
N
```

RESULT

Thus the program has been executed successfully.

Ex. No: 9

PREORDER TREE TRAVERSAL

Date:

AIM

To create a C program for Preorder traversal of the binary search tree

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Implement the function to process the preorder tree traversal.

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program

STEP 8: Stop the program.

PROGRAM

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

struct node
{
   int key;
   struct node *left;
   struct node *right;
};

//return a new node with the given value
struct node *getNode(int val)
{
   struct node *newNode;

   newNode = malloc(sizeof(struct node));

   newNode->key   = val;
   newNode->left  = NULL;
   newNode->right = NULL;

   return newNode;
}

//inserts nodes in the binary search tree
struct node *insertNode(struct node *root, int val)
{
    if(root == NULL)
        return getNode(val);

    if(root->key < val)
        root->right = insertNode(root->right,val);

    if(root->key > val)
        root->left = insertNode(root->left,val);

    return root;
}

//preorder traversal of the binary search tree
```

```
void preorder(struct node *root)
{
    if(root == NULL)
        return;

    //visit the root
    printf("%d ",root->key);

    //traverse the left subtree
    preorder(root->left);

    //traverse the right subtree
    preorder(root->right);
}

int main()
{
   struct node *root = NULL;

   int data;
    char ch;
        /*  Do while loop to display various options to select from to decide the input  */

        do
        {
            printf("\nSelect one of the operations::");
            printf("\n1. To insert a new node in the Binary Tree");
            printf("\n2. To display the nodes of the Binary Tree(via Preorder Traversal).\n
");

            int choice;
            scanf("%d",&choice);
            switch (choice)
            {
            case 1 :
                printf("\nEnter the value to be inserted\n");
                scanf("%d",&data);
                root = insertNode(root,data);
                break;
            case 2 :
                printf("\nPreorder Traversal of the Binary Tree::\n");
                preorder(root);
                break;
            default :
                printf("Wrong Entry\n");
                break;
            }

            printf("\nDo you want to continue (Type y or n)\n");
            scanf(" %c",&ch);
        } while (ch == 'Y'|| ch == 'y');

  return 0;
}
```

OUTPUT

```
Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Preorder Traversal).
1

Enter the value to be inserted
45

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Preorder Traversal).
1

Enter the value to be inserted
53

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Preorder Traversal).
1

Enter the value to be inserted
1

Do you want to continue (Type y or n)
y
```

RESULT

Thus the program has been executed successfully.

Ex. No: 10

POSTORDER TREE TRAVERSAL

Date:

AIM

To create a C program for Postorder traversal of the binary search tree

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Implement the function to process the postorder tree traversal.

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program

STEP 8: Stop the program.

PROGRAM

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

```
#include<stdlib.h>

struct node
{
    int key;
    struct node *left;
    struct node *right;
};

//return a new node with the given value
struct node *getNode(int val)
{
    struct node *newNode;

    newNode = malloc(sizeof(struct node));

    newNode->key   = val;
    newNode->left  = NULL;
    newNode->right = NULL;

    return newNode;
}
//inserts nodes in the binary search tree
struct node *insertNode(struct node *root, int val)
{
     if(root == NULL)
         return getNode(val);

     if(root->key < val)
         root->right = insertNode(root->right,val);

     if(root->key > val)
         root->left = insertNode(root->left,val);

     return root;
}

//postorder traversal of the binary search tree
void postorder(struct node *root)
{
    if(root == NULL)
```

```
        return;

    //traverse the left subtree
    postorder(root->left);

    //traverse the right subtree
    postorder(root->right);

    //visit the root
    printf("%d ",root->key);
}
int main()
{
   struct node *root = NULL;
          int data;
    char ch;

        do
        {
            printf("\nSelect one of the operations::");
            printf("\n1. To insert a new node in the Binary Tree");
            printf("\n2. To display the nodes of the Binary Tree(via Postorder Traversal).\
n");

            int choice;
            scanf("%d",&choice);
            switch (choice)
            {
            case 1 :
                printf("\nEnter the value to be inserted\n");
                scanf("%d",&data);
                root = insertNode(root,data);
                break;
            case 2 :
                printf("\nPostorder Traversal of the Binary Tree::\n");
                postorder(root);
                break;
            default :
                printf("Wrong Entry\n");
                break;
            }
```

```
         printf("\nDo you want to continue (Type y or n)\n");
         scanf(" %c",&ch);
      } while (ch == 'Y'|| ch == 'y');

  return 0;
}
```

OUTPUT

```
Select one of the operations::
```

```
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Postorder Traversal).
1

Enter the value to be inserted
12

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Postorder Traversal).
1

Enter the value to be inserted
31

Do you want to continue (Type y or n)
y

Select one of the operations::
1. To insert a new node in the Binary Tree
2. To display the nodes of the Binary Tree(via Postorder Traversal).
24
Wrong Entry

Do you want to continue (Type y or n)
y
```

RESULT

Thus the program has been executed successfully.

Ex. No: 11

SELECTION SORT

Date:

AIM

To create a C program to implement the Selection sort

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Get the input values to be sorted.

STEP 4: Implement the function to process the selection sort

STEP 5: Display the result

STEP 6: Save and compile the program

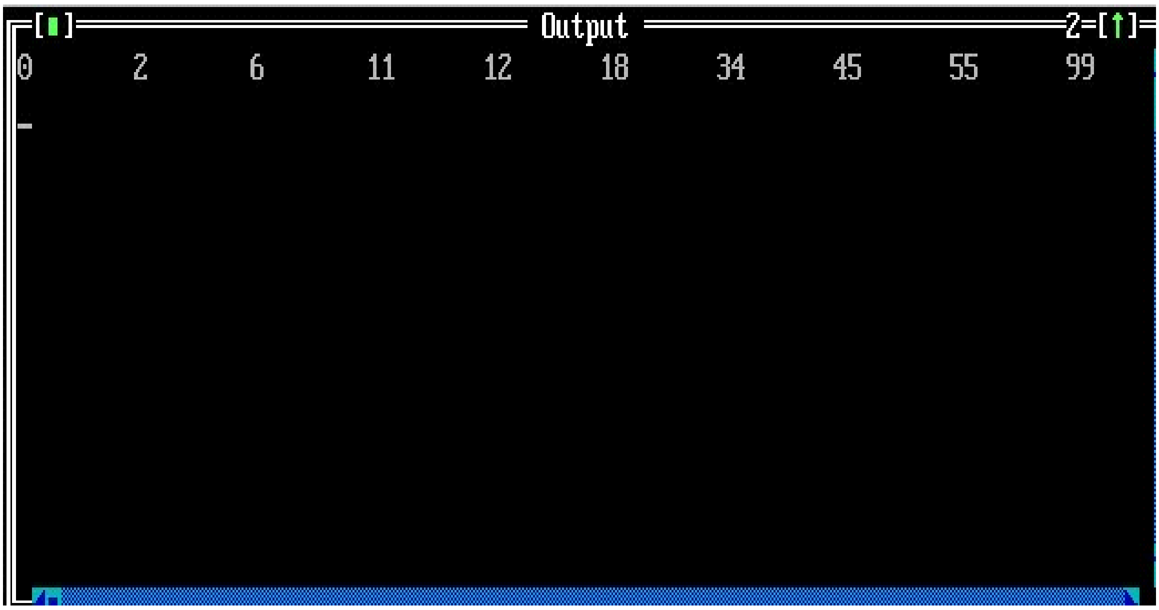
STEP 7: Run the program & Display the result.

STEP 8: Stop the program.

PROGRAM

```
#include<stdio.h>
int main()
{
int arr[10]={6,12,0,18,11,99,55,45,34,2};
int n=10;
int i,j,position,swap;
clrscr();
for(i=0;i<(n-1);i++)
{
position=i;
for(j=i+1;j<n;j++)
{
if(arr[position]>arr[j])
position=j;
}
if(position!=i)
{
swap=arr[i];
arr[i]=arr[position];
arr[position]=swap;
}
}
for(i=0;i<n;i++)
printf("%d\t",arr[i]);
getch();
return 0;
}
```

OUTPUT



```
0       2       6       11      12      18      34      45      55      99
_
```

RESULT

Thus the program has been executed successfully.

QUICK SORT

Date:

AIM

To create a C program to implement the Quick sort

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Get the input values to be sorted.

STEP 4: Implement the function to process the Quick sort

STEP 5: Display the result

STEP 6: Save and compile the program

STEP 7: Run the program & Display the result.

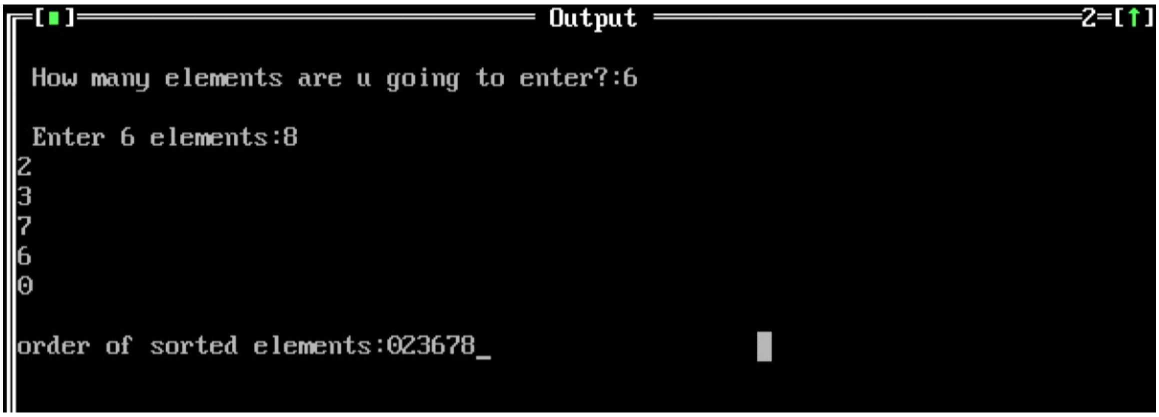
STEP 8: Stop the program.

PROGRAM

```
#include<stdio.h>
void quicksort(int number[25],int first,int last)
{
int i,j,pivot,temp;
if(first<last)
{
pivot=first;
i=first;
j=last;
while(i<j)
{
while(number[i]<=number[pivot]&&i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j)
{
temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}
}
int main()
{
```

```
int i,count,number[25];
clrscr();
printf("\n How many elements are u going to enter?:");
scanf("%d",&count);
printf("\n Enter %d elements:",count);
for(i=0;i<count;i++)
scanf("%d",&number[i]);
quicksort(number,0,count-1);
printf("\norder of sorted elements:");
for(i=0;i<count;i++)
printf("%d",number[i]);
getch();
return 0;
}
```

OUTPUT



```
 How many elements are u going to enter?:6

 Enter 6 elements:8
2
3
7
6
0

order of sorted elements:023678_
```

RESULT

Thus the program has been executed successfully.

Ex. No: 13

BUBBLE SORT

Date:

AIM

To create a C program to implement the Bubble sort

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Get the input values to be sorted.

STEP 4: Implement the function to process the Bubble sort

STEP 5: Display the result

STEP 6: Save and compile the program

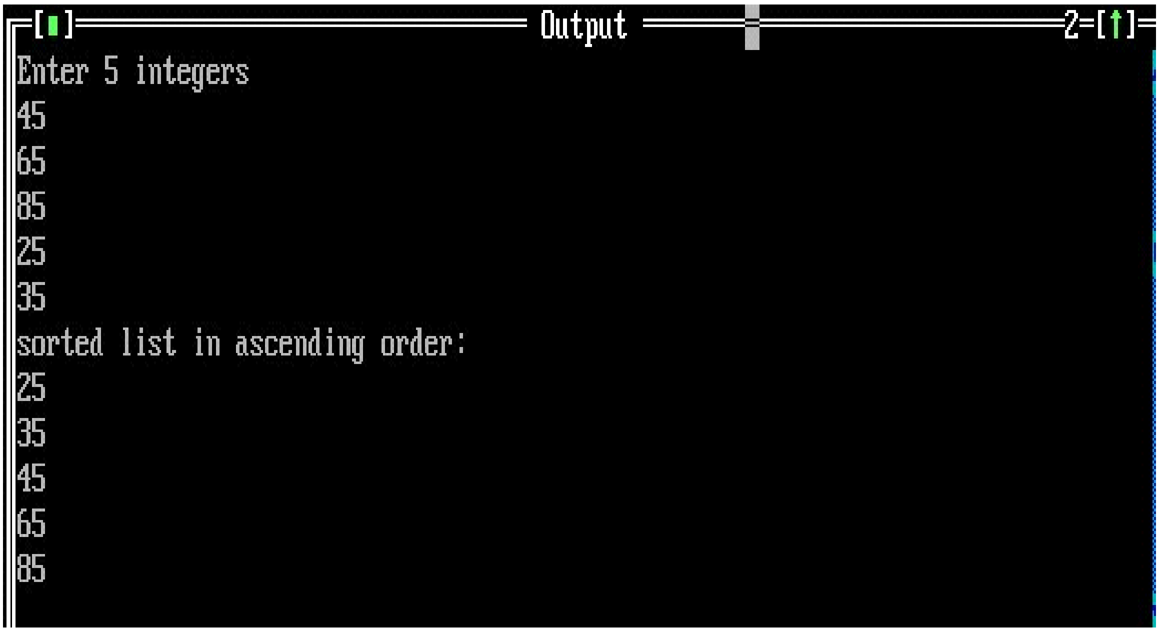
STEP 7: Run the program & Display the result.

STEP 8: Stop the program.

PROGRAM

```
#include<stdio.h>
int main()
{
int array[100],n,c,d,swap;
clrscr();
printf("\n Enter no of element:");
scanf("%d",&n);
printf("\nEnter %d integers\n",n);
for(c=0;c<n;c++)
scanf("%d",&array[c]);
for(c=0;c<n-1;c++)
{
for(d=0;d<n-c-1;d++)
{
if(array[d]>array[d+1])
{
swap=array[d];
array[d]=array[d+1];
array[d+1]=swap;
}
}
}
printf("sorted list in ascending order:\n");
for(c=0;c<n;c++)
printf("%d\n",array[c]);
getch();
return 0;
}
```

OUTPUT



```
Enter 5 integers
45
65
85
25
35
sorted list in ascending order:
25
35
45
65
85
```

RESULT

Thus the program has been executed successfully.

Ex. No: 14

MERGE SORT

Date:

AIM

To create a C program to implement the Merge sort

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Get the input values to be sorted.

STEP 4: Implement the function to process the Merge sort

STEP 5: Display the result

STEP 6: Save and compile the program

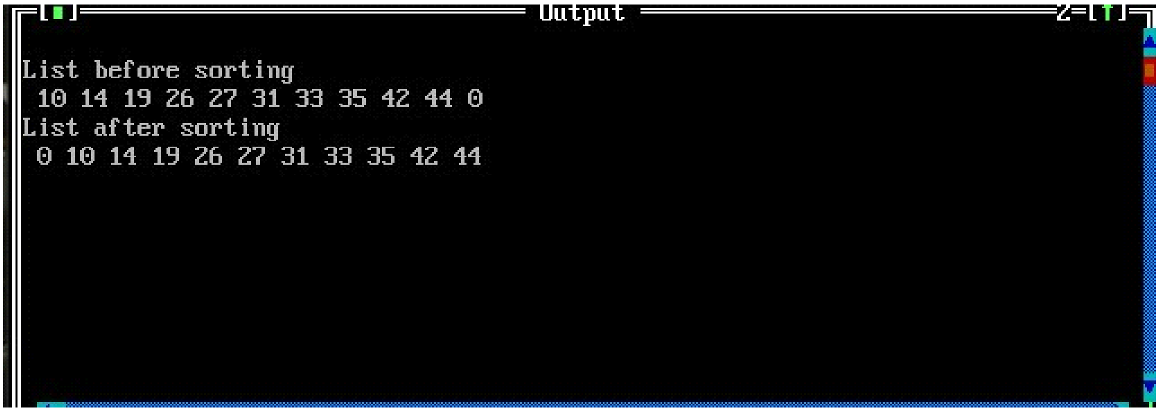
STEP 7: Run the program & Display the result.

STEP 8: Stop the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#define max 10
int a[11]={10,14,19,26,27,31,33,35,42,44,0};
int b[10];
void merging(int low,int mid,int high)
{
int l1,l2,i;
for(l1=low,l2=mid+1,i=low;l1<=mid&&l2<=high;i++)
{
if(a[l1]<=a[l2])
b[i]=a[l1++];
else
b[i]=a[l2++];
}
while(l1<=mid)
b[i++]=a[l1++];
while(l2<=high)
b[i++]=a[l2++];
for(i=low;i<=high;i++)
a[i]=b[i];
}
void sort(int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
sort(low,mid);
sort(mid+1,high);
merging(low,mid,high);
}
else
{
return;
}
}
int main()
{
int i;
clrscr();
printf("\nList before sorting\n");
for(i=0;i<=max;i++)
printf(" %d",a[i]);
sort(0,max);
printf("\nList after sorting\n");
for(i=0;i<=max;i++)
printf(" %d",a[i]);
getch();
return 0;
}
```

OUTPUT



```
List before sorting
 10 14 19 26 27 31 33 35 42 44 0
List after sorting
 0 10 14 19 26 27 31 33 35 42 44
```

RESULT

Thus the program has been executed successfully.