**PRACTICAL NO.:1(A)**

**AIM:** Write a program to store the elements in 1-D array and perform the operations like searching, sorting, reversing the elements.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

intsize,val;

void disp(int size);

void sort(int size);

void reverse(int size);

void search(intval,int size);

intarr[20];

intmain()

{

inti,ch;

printf("Enter the size of array : ");

scanf("%d",&size);

for(i=0;i<size;i++)

{

scanf("%d",&arr[i]);

}

do

{

printf("\n\*\*\*\*Main Menu\*\*\*\*\n");

printf("1.Display\n");

printf("2.Sorting\n");

printf("3.Reverse\n");

printf("Enter your Choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:disp(size);

break;

case 2:sort(size);

break;

case 3:reverse(size);

break;

case 4:printf("Enter value to be search : ");

scanf("%d",&val);

search(val,size);

break;

}

}

while(ch!=4);

getch ();

return 0;

}

void search(intval,int size)

{

inti;

for(i=0;i<size;i++)

{

if(arr[i]==val)

{

printf("Value is found at %d position.",i);

break;

}

}

if(i==size)

{

printf("Value is not found.");

}

}

void disp(int size)

{

inti;

printf("Given Array :\n");

for(i=0;i<size;i++)

{

printf("%d\n",arr[i]);

}

}

void sort(size)

{

inti,j;

for(i=0;i<size;i++)

{

for(j=0;j<size-i-1;j++)

{

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

{

int temp;

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("Sorted Array : \n");

for(i=0;i<size;i++)

{

printf("%d \n",arr[i]);

}

}

void reverse(size)

{

inti,j,temp;

j=size-1;

i=0;

while(i<j)

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

i++;

j--;

}

printf("Reverse order : \n");

for(i=0;i<size;i++)

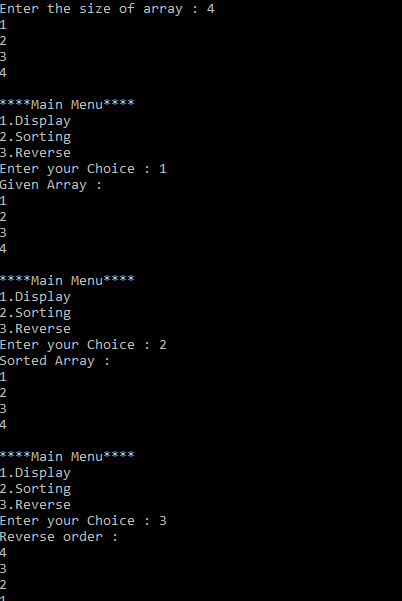
{

printf("%d \n",arr[i]);

}

}

**OUTPUT :**

****

**PRACTICAL NO.:1(B)**

**AIM :** Read the two arrays from the user and merge them and display the elements in sorted order.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

intmain()

{

int arr1[20],arr2[20],arr3[40];

inti,j,k,size1,size2,temp;

printf("Enter the array size of array 1 :");

scanf("%d",&size1);

printf("Enter the element in arra 1 :\n");

for(i=0;i<size1;i++)

{

scanf("%d",&arr1[i]);

}

printf("Enter the size of array 2 : ");

scanf("%d",&size2);

printf("Enter the element in array2 : \n");

for(j=0;j<size2;j++)

{

scanf("%d",&arr2[j]);

}

for(i=0;i<size1;i++)

{

for(j=0;j<size1-i-1;j++)

{

if(arr1[j]>arr1[j+1])

{

temp=arr1[j];

arr1[j]=arr1[j+1];

arr1[j+1]=temp;

}

}

}

for(i=0;i<size2;i++)

{

for(j=0;j<size2-i-1;j++)

{

if(arr2[j]>arr2[j+1])

{

temp=arr2[j];

arr2[j]=arr2[j+1];

arr2[j+1]=temp;

}

}

}

/\* printf("Sorting array 1\n");

for(i=0;i<size1;i++)

{

printf("%d \n",arr1[i]);

}

printf("Sorting array 2\n");

for(i=0;i<size2;i++)

{

printf("%d \n",arr2[i]);

}\*/

i=0;

j=0;

k=0;

while(i<size1 && j<size2)

{

if(arr1[i]<arr2[j])

{

arr3[k]=arr1[i];

i++;

k++;

}

else

{

arr3[k]=arr2[j];

j++;

k++;

}

}

while(i<size1)

{

arr3[k]=arr1[i];

i++;

k++;

}

while(j<size2)

{

arr3[k]=arr2[j];

j++;

k++;

}

printf("merged array \n");

for(k=0;k<size1+size2;k++)

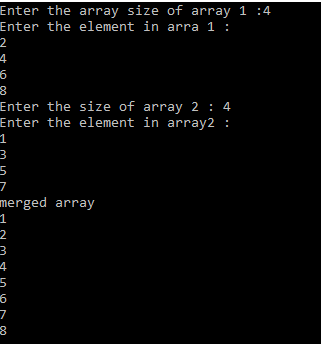
printf("%d \n",arr3[k]);

getch();

return 0;

}

**OUTPUT :**

****

**PRACTICAL NO.:1(C)**

**AIM :** write a program to perform the Matrix addition, Multiplication, Transpose operation.

**PROGRAM CODE :**

#include<stdio.h>

#include<conio.h>

int arr1[20][10],arr2[20][10],arr3[20][10];

int r1,c1,r2,c2;

void addition(int r1,int r2,int c1,int c2);

void multiplication(int r1,int r2,int c1,int c2);

void transpose1(int r1,int c1);

void transpose2(int r2,int c2);

void main()

{

inti,j,k,choice;

printf("Enter the matrix 1 size : \n");

scanf("%d %d",&r1,&c1);

printf("Enter the element in matrix 1 : \n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

scanf("%d",&arr1[i][j]);

}

printf("Matrix 1 : \n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

printf("%d \t",arr1[i][j]);

}

printf("\n");

}

printf("Enter the matrix 2 size :");

scanf("%d %d",&r2,&c2);

printf("Enter the element in matrix 2 : \n");

for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

scanf("%d",&arr2[i][j]);

}

printf("Matrix 2 : \n");

for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

{

printf("%d \t",arr2[i][j]);

}

printf("\n");

}

do

{

printf("\*\*\*\*\*\*\* Main Menu\*\*\*\*\*\*\* \n");

printf("1.Addition\n");

printf("2.Multiplication\n");

printf("3.transpose 1\n");

printf("4.transpose 2\n");

printf("Enter your chioce : ");

scanf("%d",&choice);

switch(choice)

{

case 1 : addition(r1,r2,c1,c2);

break;

case 2: multiplication(r1,r2,c1,c2);

break;

case 3: transpose1(r1,c1);

break;

case 4 : transpose2(r2,c2);

break;

}

}while(choice>0);

getch();

}

void addition(int r1,int r2,int c1,int c2)

{

inti,j;

if(r1==r2 && c1==c2)

{

printf("Addition of matrix :\n");

for(i=0;i<r1;i++)

for(j=0;j<r2;j++)

arr3[i][j]=arr1[i][j]+arr2[i][j];

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

printf("%d \t",arr3[i][j]);

}

printf("\n");

}

}

else

{

printf("ORDER INCORRECT. \n");

}

}

void multiplication(int r1,int r2,int c1,int c2)

{

inti,j,k;

if(c1==r2)

{

printf("Multiplication of matrix :\n");

for(i=0;i<r1;i++)

{

for(j=0;j<c2;j++)

{

arr3[i][j]=0;

int k;

for(k=0;k<r1;k++)

{

arr3[i][j]+=arr1[i][k]\*arr2[k][j];

}

printf("%d\t",arr3[i][j]);

} printf("\n");

}

}

else

{

printf("ORDER INCORRECT.\n");

}

}

void transpose1(int r1,int c1)

{

inti,j;

printf("Transpose matrix 1 \n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

arr3[j][i]=arr1[i][j];

}

}

for(i=0;i<c1;i++)

{

for(j=0;j<r1;j++)

{

printf("%d\t",arr3[i][j]);

}

printf("\n");

}

}

void transpose2(int r2,int c2)

{

inti,j;

printf("Transpose matrix 2 \n");

for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

{

arr3[j][i]=arr1[i][j];

}

}

for(i=0;i<c2;i++)

{

for(j=0;j<r2;j++)

{

printf("%d\t",arr3[i][j]);

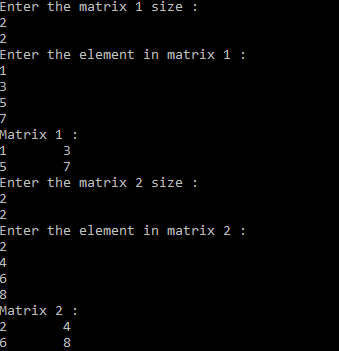
}

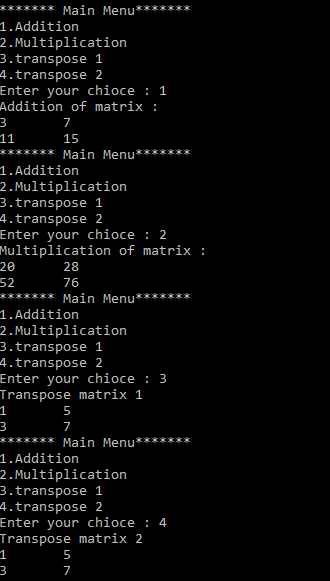
printf("\n");

}

}

**OUTPUT :**

****

****

**PRACTICAL NO.:2(A)**

**AIM :**  Write a program to create a single linked list and display the node elements in reserve order.

**Pogram code:**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*next;

};

struct node \*start=NULL;

struct node \*create(struct node \*start);

struct node \*dispaly(struct node \*start);

void reverse(struct node \*start);

int main()

{

clrscr();

start=create(start);

start=dispaly(start);

printf("\n");

printf("Reverse \t");

reverse(start);

getch ();

return 0;

}

struct node \*create(struct node \*start)

{

struct node \*new\_node=NULL,\*temp=NULL;

int val;

printf("Enter -1 value to exit list.\n");

printf("Enter the value : \n");

scanf("%d",&val);

while(val!=-1)

{

new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->info=val;

if(start==NULL)

{

start=new\_node;

new\_node->next=NULL;

}

else

{

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=new\_node;

new\_node->next=NULL;

}

printf("Enter the value : \n");

scanf("%d",&val);

}

printf("List is successfully created.\n");

return start;

}

struct node \*dispaly(struct node \*start)

{

struct node \*temp=NULL;

temp=start;

printf("List is :\n");

while(temp!=NULL)

{

printf("%d \t",temp->info);

temp=temp->next;

}

return start;

}

void reverse(struct node \*start)

{

struct node \*prev=NULL;

struct node \*current=start;

struct node \*next\_node;

while(current!=NULL)

{

next\_node=current->next;

current->next=prev;

prev=current;

current=next\_node;

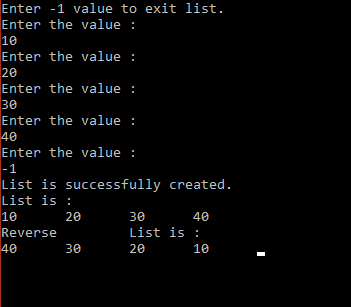
}

start=prev;

start=dispaly(start);

}

**OUTPUT :**

****

**PRACTICAL NO.:2(B)**

**AIM :** Write a program to search the elements in the linked list and display the same.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*next;

};

struct node \*start=NULL;

struct node \*create(struct node \*start);

struct node \*dispaly(struct node \*start);

struct node \*search(struct node \*start);

int main()

{

start=create(start);

start=dispaly(start);

printf("\n");

start=search(start);

getch ();

return 0;

}

struct node \*create(struct node \*start)

{

struct node \*new\_node=NULL,\*temp=NULL;

int val;

printf("Enter -1 value to exit list.\n");

printf("Enter the value : \n");

scanf("%d",&val);

while(val!=-1)

{

new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->info=val;

if(start==NULL)

{

start=new\_node;

new\_node->next=NULL;

}

else

{

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=new\_node;

new\_node->next=NULL;

}

printf("Enter the value : \n");

scanf("%d",&val);

}

printf("List is successfully created.\n");

return start;

}

struct node \*dispaly(struct node \*start)

{

struct node \*temp=NULL;

temp=start;

printf("List is :\n");

while(temp!=NULL)

{

printf("%d \t",temp->info);

temp=temp->next;

}

return start;

}

struct node \*search(struct node \*start)

{

int val,count;

struct node \*temp;

printf("\nwhich value are you looking for?\n");

scanf("%d",&val);

count=1;

temp=start;

while(temp->info!=val && temp->next!=NULL)

{

temp=temp->next;

count++;

}

//temp=temp->next;

if(temp->next==NULL && temp->info!=val)

{

printf("value not found");

}

else if(temp->next==NULL && temp->info==val)

{

printf("value found at %d node",count);

}

else

{

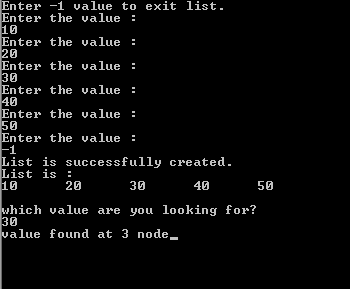
printf("value found at %d node",count);

}

return start;

}

**OUTPUT :**



**PRACTICAL NO:-2(C)**

**AIM:**Write a program to create double linked list and sort the elements in the linked list.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<stdlib.h>

struct node

{int data;

struct node \*next;

struct node \*prev;

};

struct node \*start=NULL;

struct node \*create(struct node \*start);

struct node \*display(struct node \*start);

struct node \*sort(struct node \*start);

int main()

{

start=create(start);

start=display(start);

printf("\n");

printf("sort \t");

start=sort(start);

}

struct node \*create(struct node \*start)

{

struct node \*new\_node=NULL,\*temp=NULL,prev;

int val;

printf("Enter the data or enter -1 to exit:");

scanf("%d",&val);

while(val!=-1)

{

new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->data=val;

if(start==NULL)

{

start=new\_node;

new\_node->next=NULL;

new\_node->prev=NULL;

}

else

{

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=new\_node;

new\_node->prev=NULL;

new\_node->next=NULL;

}

printf("enter the data or enter -1 to exit:");

scanf("%d",&val);

}

printf("Linked list successfully created.\n");

return start;

}

struct node \*display(struct node \*start)

{

struct node \*temp=NULL;

temp=start;

printf("\nThe Linked list is:");

while(temp->next!=NULL)

{

printf("\t %d \t",temp->data);

temp=temp->next;

}

if(temp->next==NULL)

printf("%d \n",temp->data);

printf("\n");

return start;

}

struct node \*sort(struct node\*start)

{

struct node \*temp1=start;

struct node \*temp2,\*temp;

int x;

while (temp1->next!=NULL)

{

temp2=start;

while(temp2->next!=NULL)

{

temp=temp2->next;

if(temp2->data>temp->data)

{

x=temp->data;

temp->data=temp2->data;

temp2->data=x;

}

temp2=temp2->next;

}

temp1=temp1->next;

}

temp=start;

printf("The Linked List is:");

while(temp->next!=NULL)

{

printf("%d \t",temp->data);

temp=temp->next;

}

if(temp->next==NULL)

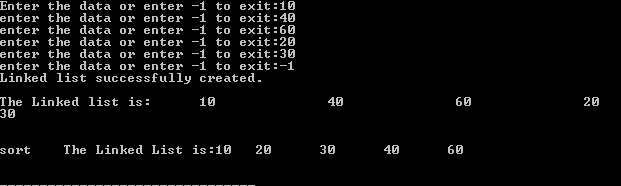
printf("%d \n",temp->data);

printf("\n");

return start;

}

**OUTPUT :**

****

**PRACTICAL NO:-3(A)**

**AIM :**  Write a program to implement the concept of stack with Push, Pop, Display and exit operation.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#define MAX 30

int stack[MAX];

int top =-1; //Stack is empty.

void push();

int pop();

int peek();

void display();

int main()

{

int choice;

do

{

printf("\n \*\*\*\* Main Menu \*\*\*\* \n");

printf("1.Push\n");

printf("2.Pop \n");

printf("3.Peek \n");

printf("4.Display \n");

printf("Enter your choice :");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1: push();

break;

case 2 : pop();

break;

case 3 : peek();

break;

case 4 : display();

break;

case 5 : break;

}

}

while(choice!=5);

return 0;

}

void push()

{

int val;

if(top == MAX -1)

{

printf("Stack is full.");

} else

{

printf("Enter the value to be pushed : ");

scanf("%d",&val);

stack[++top]=val;

printf("Successfully pushed.\n");

}

}

int pop()

{

if(top == -1)

{

printf("Stack is already empty.");

}

else

{

int val = stack[top];

top--;

printf("The value is popped : %d",val);

}

}

int peek()

{

if(top == -1)

{

printf("Stack is empty.");

}

else

{

int topmost = stack[top];

printf("The topmost element of stack : %d ",topmost);

}

}

void display()

{

if(top == -1)

{

printf("Stack is empty.");

}

else

{

int i;

printf("Stack is : ");

for(i=top;i>=0;i--)

{

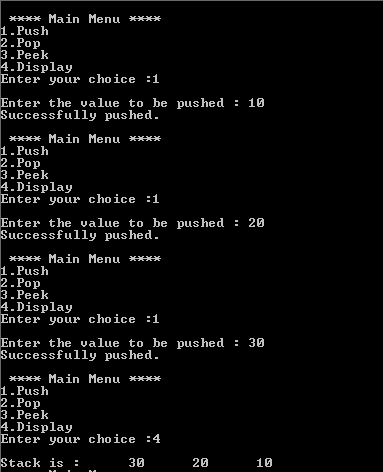
printf("\t%d",stack[i]);

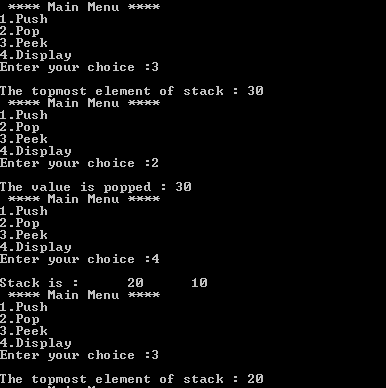
}

}

}

**OUTPUT :**

****

****

**PRACTICAL NO.:3(B)**

**AIM :** Write a program to implement Tower of Hanio problem .

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

int move(int n,char source,char temp,char destination);

int main()

{

int n;

printf("enter number of Disk");

scanf("%d",&n);

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

getch();

return 0;

}

int move(int n,char source,char temp,char destination)

{

if(n == 1)

{

printf("\n Move from %c to %c",source,destination);

}

else

{

move(n-1,source,destination,temp);

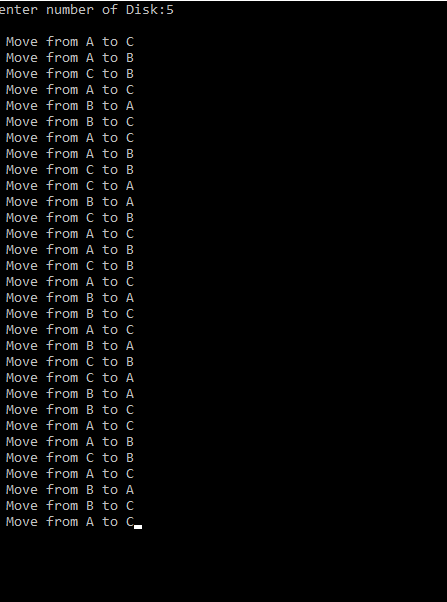
move(1,source,temp,destination);

move(n-1,temp,source,destination);

}

}

**OUTPUT :**



**PRACTICAL NO:-4(A)**

**AIM :** Write a program to implement the concept of Queue with Insert, Delete, Display and exit operation.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#define max 30

int rear=-1;

int front=-1;

void insert();

int deleteq();

void display();

int q[max];

void insert()

{

int val;

printf("Enter value to be inserted :");

scanf("%d",&val);

if(rear==max-1)

{

printf("Queue is full.");

}

else if(front==-1)

{

front=rear=0;

q[rear]=val;

printf("Value inserted successfully.");

}

else

{

q[++rear]=val;

printf("Value inserted successfully.");

}

}

int deleteq()

{

if(front==-1)

{

printf("Queue is already empty.");

return -1;

}

else if(front==rear) //Only one item is present.

{ int val ;

val=q[front];

front=rear=-1;

printf("Value to be deletede : %d",val);

return val;

}

else

{

int val ;

val=q[front];

front++;

printf("Value to be deletede : %d",val);

return val;

}

}

void display()

{

if(front==-1)

{

printf("Queue is empty.");

}

else

{

int i;

printf("Queue is :");

for(i=front;i<=rear;i++)

{

printf("%d",q[i]);

}

}

}

int main()

{

int choice;

do

{

printf("\n \*\*\*\* Main Menu \*\*\*\* \n");

printf("1.Insert\n");

printf("2.Delete\n");

printf("3.Display \n");

printf("Enter your choice :");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1: insert();

break;

case 2 : deleteq();

break;

case 3 : display();

break;

case 4 : break;

}

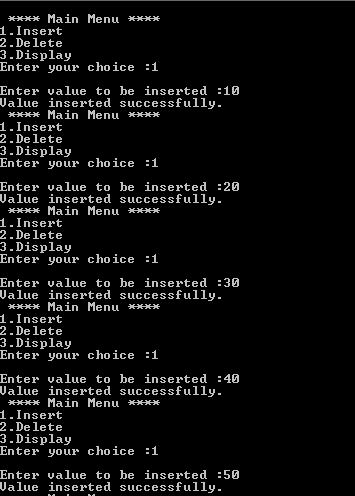
}

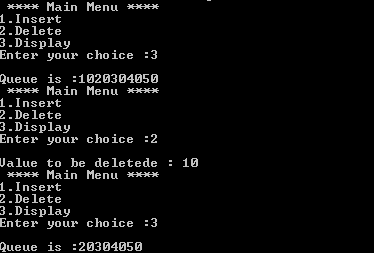
while(choice!=4);

return 0;

}

**OUTPUT :**

****

****

**PRACTICAL NO:-4(B)**

**AIM :** Write a program to implement the concept of circular Queue.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#define max 5

int front = -1;

int rear = -1;

void insetr();

void display();

intdeleteq();

int q[max];

void insert()

{

intval;

printf("Enter value :");

scanf("%d",&val);

if((rear+1)%max==front)

{

printf("Queue is full.");

}

else if(rear==-1)

{

rear=front=0;

q[rear]=val;

printf("Inserted successfully.");

}

Else

{

rear=(rear + 1)%max;

q[rear]=val;

printf("Inserted successfully.");

}

}

intdeleteq()

{

intval;

if(front== -1)

{

printf("Queue is empty.");

return -1;

}

else if(front == rear)

{

intval=q[front];

front=rear= -1;

printf("Deleted value : %d",val);

return val;

}

else

{

val=q[front];

front=(front+1)%max;

printf("Deleted value : %d",val);

return val;

}

}

void display()

{

inti;

if(front ==-1)

{

printf("Queue is empty.");

}

else

{

printf("Queue is :");

for(i=front;i!=rear;i=(i+1)%max)

{

printf("%d",q[i]);

}

printf("%d",q[i]);

}

}

intmain()

{ int choice;

clrscr();

do

{

printf("\n \*\*\*\* Main Menu \*\*\*\* \n");

printf("1.Insert\n");

printf("2.Delete\n");

printf("3.Display \n");

printf("Enter your choice :");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1: insert();

break;

case 2 :deleteq();

break;

case 3 : display();

break;

case 4 : break;

}

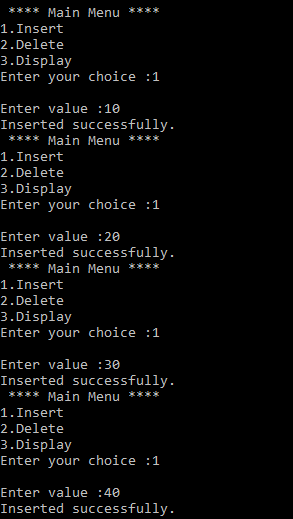
}

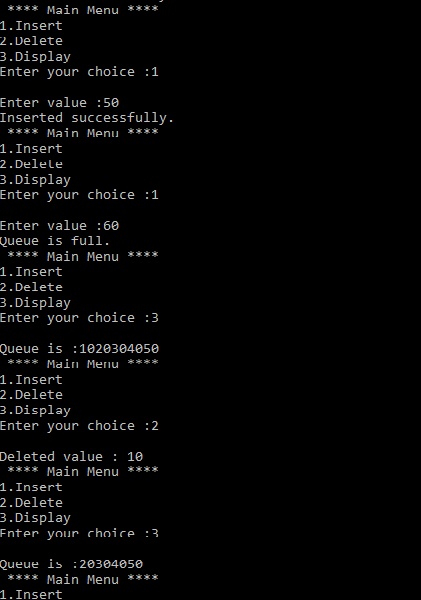
while(choice!=4);

return 0;

}

**OUTPUT :**

****

****

**PRACTICAL NO:-4(C)**

**AIM :** Write a program to implement the concept of Deque.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#define max 5

int front = -1;

int rear = -1;

intinsert\_rear();

intinsert\_front();

void display();

intdeleteq\_rear();

intdeleteq\_front();

int q[max];

intmain()

{

int choice;

clrscr();

do

{

printf("\n \*\*\*\* Main Menu \*\*\*\* \n");

printf("1.Insert From Rear\n");

printf("2.Insert From Front\n");

printf("3.Delete From Front \n");

printf("4.Delete From Rear \n");

printf("5.Display\n");

printf("Enter your choice :");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1: insert\_rear();

break;

case 2 :insert\_front();

break;

case 3 :deleteq\_front();

break;

case 4 :deleteq\_rear();

break;

case 5 : display();

break;

case 6 : break;

}

}

while(choice!=6);

return 0;

}

intinsert\_rear()

{ intval;

printf("Enter value :");

scanf("%d",&val);

if((rear+1)%max==front)

{

printf("Queue is full.");

return 0;

}

else if(rear==-1)

{

rear=front=0;

q[rear]=val;

printf("Inserted successfully.");

return val;

}

else

{

rear=(rear + 1)%max;

q[rear]=val;

printf("Inserted successfully.");

return val;

}

}

intinsert\_front()

{ intval;

printf("Enter value :");

scanf("%d",&val);

if((rear+1)%max==front)

{

printf("Queue is full.");

return 0;

}

Else if(front==-1)

{

rear=front=0;

q[front]=val;

printf("Inserted successfully.");

return val;

}

else

{

front=(front-1+max)%max;

q[front]=val;

printf("Inserted successfully.");

return val;

}}

intdeleteq\_front()

{ intval;

if(front== -1)

{

printf("Queue is empty.");

//return -1;

}

else if(front == rear)

{

intval=q[front];

front=rear= -1;

printf("Deleted value : %d",val);

return val;

}

else

{

val=q[front];

front=(front+1)%max;

printf("Deleted value : %d",val);

return val;

}}

intdeleteq\_rear()

{ intval;

if(rear== -1)

{

printf("Queue is empty.");

return -1;

}

else if(front == rear)

{

intval=q[rear];

front=rear= -1;

printf("Deleted value : %d",val);

return val;

}

else

{

val=q[rear];

rear=(rear-1+max)%max;

printf("Deleted value : %d",val);

return val;

}}

void display()

{

inti;

if(front ==-1)

{

printf("Queue is empty.");

}

else

{

printf("Queue is :");

for(i=front;i!=rear;i=(i+1)%max)

{

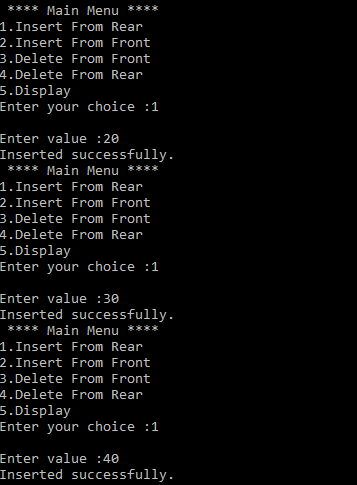
printf(" %d ",q[i]);

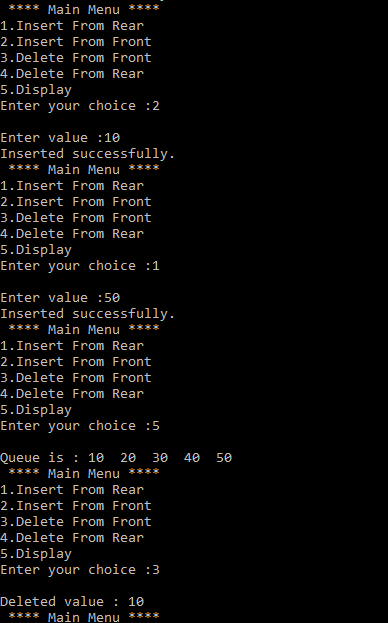
}

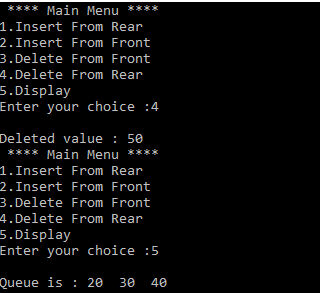
printf(" %d ",q[i]);

}}

**OUTPUT :**

****

****

****

**PRACTICAL NO:-5(A)**

**AIM :** Write a program to implement bubble sort.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

int size,val;

void disp(int size);

int sort(int size);

int arr[20];

int main()

{

int i,ch;

printf("Enter the size of array : ");

scanf("%d",&size);

for(i=0;i<size;i++)

{

scanf("%d",&arr[i]);

}

do

{

printf("\n\*\*\*\*Main Menu\*\*\*\*\n");

printf("1.Display\n");

printf("2.Sorting\n");

printf("Enter your Choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:disp(size);

break;

case 2:sort(size);

break;

}

}

while(ch!=2);

getch ();

return 0;

}

void disp(int size)

{

int i;

printf("Given Array :\n");

for(i=0;i<size;i++)

{

printf("%d\n",arr[i]);

}

}

int sort(size)

{

int i,j;

for(i=0;i<size;i++)

{

for(j=0;j<size-i-1;j++)

{

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

{

int temp;

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("Sorted Array : \n");

for(i=0;i<size;i++)

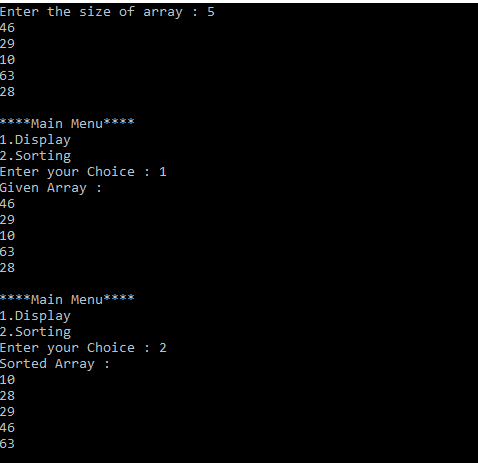
{

printf("%d \n",arr[i]);

}

}

**OUTPUT :**

****

**PRACTICAL NO:-5(B)**

**AIM :** Write a program to implement selection sort.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

int selection\_sort(int n);

int A[20];

int selection\_sort(int n)

{

int imin,i,j,temp;

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

{

imin=i;

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

{

if(A[imin]>A[j])

{

imin=j;

}

}

temp = A[i];

A[i] = A[imin];

A[imin] = temp;

}

printf("Successfully sorted using Selection sort :");

}

int main()

{

int n,i;

printf("Enter the size :");

scanf("%d",&n);

printf("Enter the element :\n");

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

{

scanf("%d",&A[i]);

printf("\n");

}

selection\_sort(n);

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

{

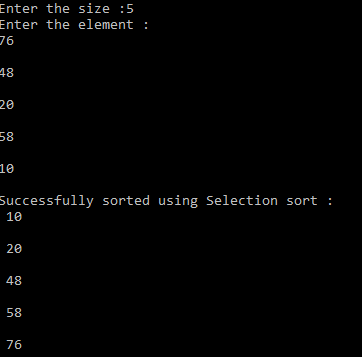
printf("\n %d\n",A[i]);

}

return 0;

}

**OUTPUT :**

****

**PRACTICAL NO:-5(C)**

**AIM :** Write a program to implement insertionsort.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

int A[10];

void insertion\_sort(int n)

{

int val,vacant,i;

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

{

val=A[i];

vacant=i;

while(A[vacant-1]>val && vacant!=0)

{

A[vacant]=A[vacant-1];

vacant=vacant - 1;

}

A[vacant]=val;

}

printf("Successfully sorted using Insertion Sort Algorithm : \n");

}

void main()

{

int n,i;

printf("Enter the size of array : ");

scanf("%d",&n);

printf("Enter the elements :\n");

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

{

scanf("%d",&A[i]);

printf("\n");

}

insertion\_sort(n);

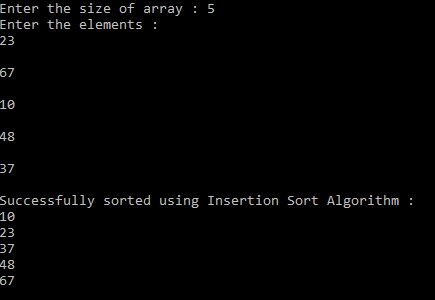
for(i=0;i<n;i++)

{

printf("%d \n",A[i]);

}}

**OUTPUT :**

****

**PRACTICAL NO.:6(A)**

**AIM :** Write a program to implement merge sort.

**PROGRAM CODE:**

#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("List before sorting\n");

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

printf("%d ", a[i]);

sort(0, max);

printf("\nList after sorting\n");

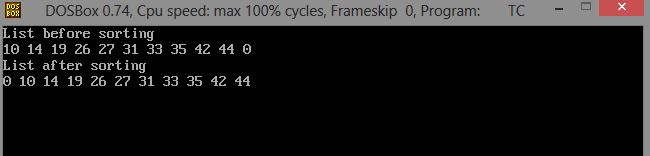
for(i = 0; i <= max; i++)

printf("%d ", a[i]);

getch();

}

**OUTPUT :**

****

**PRACTICAL NO.:6(B)**

**AIM:** Write a program to search the element using seqential search.

**PROGRAM CODE:**

#include<stdio.h>

#include<conio.h>

intsize,val;

void disp(int size);

void search(intval,int size);

intarr[20];

intmain()

{

inti,ch;

printf("Enter the size of array : ");

scanf("%d",&size);

for(i=0;i<size;i++)

{

scanf("%d",&arr[i]);

}

do

{

printf("\n\*\*\*\*Main Menu\*\*\*\*\n");

printf("1.Display\n");

printf("2.Search\n");

printf("Enter your Choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:disp(size);

break;

case 2:printf("Enter value to be search : ");

scanf("%d",&val);

search(val,size);

break;

}

}

while(ch!=2);

getch ();

return 0;

}

void search(intval,int size)

{

inti;

for(i=0;i<size;i++)

{

if(arr[i]==val)

{

printf("Value is found at %d position.",i);

break;

}

}

if(i==size)

{

printf("Value is not found.");

}

}

void disp(int size)

{

inti;

printf("Given Array :\n");

for(i=0;i<size;i++)

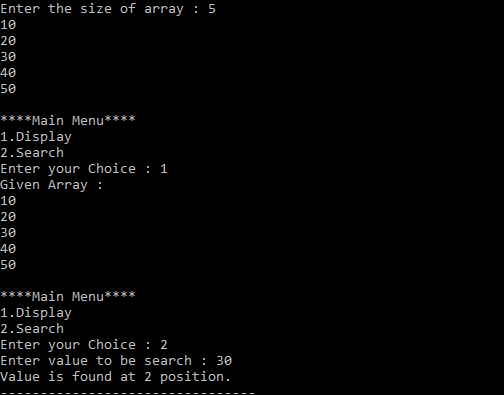
{

printf("%d\n",arr[i]);

}

}

**OUTPUT :**

****

**PRACTICAL NO.:6(C)**

**AIM :** Write a program to search the element using binary search.

**PROGRAM CODE:**

#include <stdio.h>

intmain()

{

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

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

scanf("%d",&n);

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

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

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

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

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last)

{

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search)

{

printf("%d found at location %d.\n", search, middle+1);

break;

}

else if(array[middle]>search)

last = middle - 1;

middle = (first + last)/2;

}

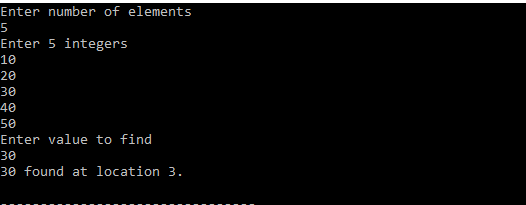
if (first > last)

printf("Not found! %d is not present in the list.\n", search);

return 0;

}

**OUTPUT :**

****

**PRACTICAL NO.:7**

**(**Implement the following data structure techiniques**)**

**Aim 7(A):** Write a program to create the tree and display the element.

**Aim 7(B):** Write a program to construct the binary tree.

**Aim 7(C):** Write a program for inorder,postorder and preorder traversal of tree.

**PROGRAM CODE:**

#include<stdio.h>

#include<malloc.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

struct node \*root=NULL;

struct node \*create(struct node\*);

struct node \*display(struct node\*);

void preorder(struct node \*temp);

void postorder(struct node \*temp);

void inorder(struct node \*temp);

intmain()

{

intchoice,val,count,min,max;

do

{

printf("\*\*\*\* Main Menu \*\*\*\n");

printf("1. create a binary search\n");

printf("2. Display the tree \n");

printf("3. EXIT \n");

printf("Enter your choice:");

scanf("%d",&choice);

printf("\n\n");

switch(choice)

{

case 1:root=create(root);

break;

case 2:root=display(root);

break;

case 3:break;

}

}while(choice!=3);

return 0;

}

struct node \*create(struct node \*root)

{

struct node \*newnode=NULL,\*temp=NULL,\*parent=NULL;

intval;

printf("Enter the data or enter -1 to exit:");

scanf("%d",&val);

while(val!=-1)

{

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=val;

if(root==NULL)

{

root=newnode;

newnode->left=NULL;

newnode->right=NULL;

}

else

{

temp=root;

while(temp!=NULL)

{

parent=temp;

if(val<temp->data)

{

temp=temp->left;

}

else

{

temp=temp->right;

}

}

if(val<parent->data)

{

parent->left=newnode;

newnode->left=NULL;

newnode->right=NULL;

}

else

{

parent->right=newnode;

newnode->left=NULL;

newnode->right=NULL;

}

}

printf("Enter the data or enter -1 to exit:");

scanf("%d",&val);

}

printf("Succesfully created \n");

return root;

}

struct node \*display(struct node \*root)

{

int choice1;

printf("\*\*\* Display Menu\*\*\*\n");

printf("1.pre-order\n");

printf("2.In-order\n");

printf("3.post-order\n");

printf("4. EXIT\n");

printf("Enter your choice :");

scanf("%d",&choice1);

switch(choice1)

{

case 1:printf("\tThe Pre-order Traveral is:");

preorder(root);

break;

case 2:printf("\tThe in order traversal is:");

inorder(root);

break;

case 3:printf("\tThe post-order traversal is:");

postorder(root);

break;

case 4:break;

}

printf("\n");

return root;

}

void preorder(struct node \*temp)

{

if(temp!=NULL)

{

printf("%d",temp->data);

preorder(temp->left);

preorder(temp->right);

}

}

void postorder(struct node \*temp)

{

if(temp!=NULL)

{

postorder(temp->left);

postorder(temp->right);

printf("%d",temp->data);

}

}

void inorder(struct node \*temp)

{

if(temp!=NULL)

{

inorder(temp->left);

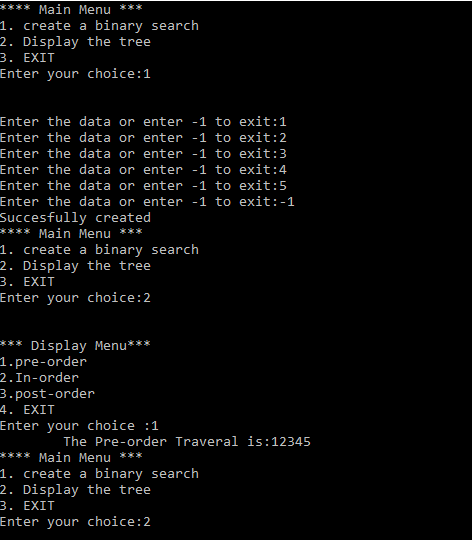
printf("%d",temp->data);

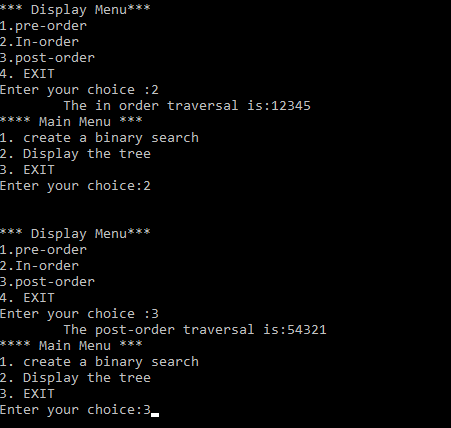
inorder(temp->right);

}

}

**OUTPUT :**

****

****

**PRACTICAL NO.:8(A)**

**AIM :** Write a program to insert the element into maximum heap.

**PROGRAM CODE:**

#include<stdio.h>

#include<string.h>

#define SIZE 30

int a[SIZE],n;

void maxheapify(int a[],int i,int n1);

void buildheap(int a[],int n1);

void heap\_sort(int a[]);

void swap(int i,int j);

int length(int a[]);

int main()

{

int i,j;

printf("Enter the number of element:");

scanf("%d",&n);

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

{

printf("Enter a value:");

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

buildheap(a,i);

}

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

{

printf("%d",a[j]);

}

printf("\n");

}

void buildheap(int a[],int n1)

{

int i,j;

for(i=(n1/2)-1;i>=0;i--)

{

maxheapify(a,i,n1);

}

for(j=0;j<n1;j++)

{

printf("%d",a[j]);

}

printf("\n\n");

}

void maxheapify(int a[],int i,int n1)

{

int max,l,r;

max=i;

l=2\*i+1;

r=2\*i+2;

if(l<n1&&r<n1)

{

if(a[l]>a[max])

{

max=l;

}

if(a[r]>a[max])

{

max=r;

}

}

else if(l<n1&&r>=n1)

{

if(a[l]>a[max])

{

max=l;

}

}

else if(l>=n1&&r<n1)

{

if(a[r]>a[max])

{

max=r;

}

}

if(i!=max)

{

swap(i,max);

maxheapify(a,max,n1);

}

}

void swap(int i,int j)

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

int length(int a[])

{

int i=0;

while(a[i]!='\0')

{

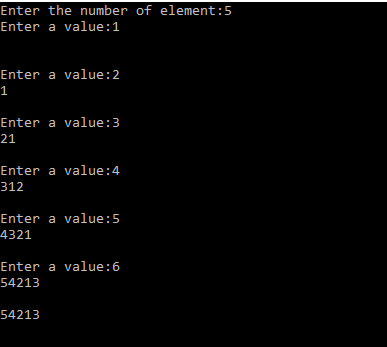
i++;

}

return i;

}

**OUTPUT :**



**PRACTICAL NO.:8(B)**

**AIM :** Write a program to insert the element into minimum heap.

**PROGRAM CODE:**

#include<stdio.h>

#include<string.h>

#define SIZE 30

int a[SIZE],n;

void maxheapify(int a[],inti,int n1);

void buildheap(int a[],int n1);

void heap\_sort(int a[]);

void swap(inti,int j);

intlength(int a[]);

intmain()

{

inti,j;

printf("Enter the number of element:");

scanf("%d",&n);

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

{

printf("Enter a value:");

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

buildheap(a,i);

}

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

{

printf("%d",a[j]);

}

printf("\n");

}

void buildheap(int a[],int n1)

{

inti,j;

for(i=(n1/2)-1;i>=0;i--)

{

maxheapify(a,i,n1);

}

for(j=0;j<n1;j++)

{

printf("%d",a[j]);

}

printf("\n");

}

void maxheapify(int a[],inti,int n1)

{

intmin,l,r;

min=i;

l=2\*i+1;

r=2\*i+2;

if(l<n1&&r<n1)

{

if(a[l]<a[min])

{

min=l;

}

if(a[r]<a[min])

{

min=r;

}

}

else if(l<n1&&r>=n1)

{

if(a[l]<a[min])

{

min=l;

}

}

else if(l>=n1&&r<n1)

{

if(a[r]<a[min])

{

min=r;

}

}

if(i!=min)

{

swap(i,min);

maxheapify(a,min,n1);

}

}

void swap(inti,int j)

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

intlength(int a[])

{

inti=0;

while(a[i]!='\0')

{

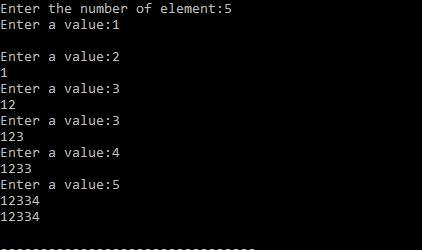
i++;

}

return i;

}

**OUTPUT :**

****