**4. Write a C++ program to implement a double ended queue ADT using an array?**

#include<iostream>

using namespace std;

#define MAX 7

class deque

{

int deque\_arr[MAX];

int front;

int rear;

public:

deque()

{

front=rear=-1;

}

void insert\_frontEnd(int item);

void insert\_rearEnd(int item);

int delete\_frontEnd();

int delete\_rearEnd();

void display();

int isEmpty();

int isFull();

};

int main()

{

int choice,item;

deque ob;

// clrscr();

cout<<"\n\n1.Insert at the front end\n";

cout<<"2.Insert at the rear end\n";

cout<<"3.Delete from front end\n";

cout<<"4.Delete from rear end\n";

cout<<"5.Display\n";

cout<<"6.Quit\n";

while(1)

{

cout<<"\nEnter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"\nInput the element for adding in queue : ";

cin>>item;

ob.insert\_frontEnd(item);

break;

case 2:

cout<<"\nInput the element for adding in queue : ";

cin>>item;

ob.insert\_rearEnd(item);

break;

case 3:

cout<<"\nElement deleted from front end is :\n",ob.delete\_frontEnd();

break;

case 4:

cout<<"\nElement deleted from rear end is : \n",ob.delete\_rearEnd();

break;

case 5:

ob.display();

break;

case 6:

exit(0);

default:

cout<<"\nWrong choice\n";

}/\*End of switch\*/

}/\*End of while\*/

return 0;

}/\*End of main()\*/

void deque::insert\_frontEnd(int item)

{

if( isFull() )

{

cout<<"\nQueue Overflow\n";

return;

}

if( front==-1 )/\*If queue is initially empty\*/

{

front=0;

rear=0;

}

else if(front==0)

front=MAX-1;

else

front=front-1;

deque\_arr[front]=item ;

}/\*End of insert\_frontEnd()\*/

void deque::insert\_rearEnd(int item)

{

if( isFull() )

{

cout<<"\nQueue Overflow\n";

return;

}

if(front==-1) /\*if queue is initially empty\*/

{

front=0;

rear=0;

}

else if(rear==MAX-1) /\*rear is at last position of queue \*/

rear=0;

else

rear=rear+1;

deque\_arr[rear]=item ;

}/\*End of insert\_rearEnd()\*/

int deque::delete\_frontEnd()

{

int item;

if( isEmpty() )

{

cout<<"\nQueue Underflow\n";

exit(1);

}

item=deque\_arr[front];

if(front==rear) /\*Queue has only one element \*/

{

front=-1;

rear=-1;

}

else

if(front==MAX-1)

front=0;

else

front=front+1;

return item;

}/\*End of delete\_frontEnd()\*/

int deque::delete\_rearEnd()

{

int item;

if( isEmpty() )

{

cout<<"\nQueue Underflow\n";

exit(1);

}

item=deque\_arr[rear];

if(front==rear) /\*queue has only one element\*/

{

front=-1;

rear=-1;

}

else if(rear==0)

rear=MAX-1;

else

rear=rear-1;

return item;

}/\*End of delete\_rearEnd() \*/

int deque::isFull()

{

if ( (front==0 && rear==MAX-1) || (front==rear+1) )

return 1;

else

return 0;

}/\*End of isFull()\*/

int deque::isEmpty()

{

if( front == -1)

return 1;

else

return 0;

}/\*End of isEmpty()\*/

void deque::display()

{

int i;

if( isEmpty() )

{

cout<<"\nQueue is empty\n";

return;

}

cout<<"\nQueue elements :\n";

i=front;

if( front<=rear )

{

while(i<=rear)

cout<<deque\_arr[i++]<<" ";

}

else

{

while(i<=MAX-1)

cout<<deque\_arr[i++]<<" ";

i=0;

while(i<=rear)

cout<<deque\_arr[i++]<<" ";

}

cout<<"\n";

}/\*End of display() \*/