Assignment 7:

A double-ended queue (deque) is a linear list in which additions and deletions may be

made at either end. Obtain a data representation mapping a deque into a one-

dimensional array. Write C++ program to simulate deque with functions to add and

delete elements from either end of the deque.

|  |
| --- |
| /\*-----------------------------------------------------  Assignment 7:  A double-ended queue (deque) is a linear list in which additions and deletions may be  made at either end. Obtain a data representation mapping a deque into a one-  dimensional array. Write C++ program to simulate deque with functions to add and  delete elements from either end of the deque.  -----------------------------------------------------\*/  #include <bits/stdc++.h>  using namespace std;  struct Node  {  int data;  Node \*prev, \*next;  static Node\* getnode(int data)  {  Node\* newNode = (Node\*)malloc(sizeof(Node));  newNode->data = data;  newNode->prev = newNode->next = NULL;  return newNode;  }  };  class Deque  {  Node\* begin;  Node\* rear;  public:  Deque()  {  begin = rear = NULL;  }  void insert\_begin(int data);  void insert\_rear(int data);  void delete\_begin();  void delete\_rear();  int display();  };  void Deque::insert\_begin(int data)  {  Node\* newNode = Node::getnode(data);  if (newNode == NULL)  cout << "OverFlow\n";  else  {  if (begin == NULL)  rear = begin = newNode;  else  {  newNode->next = begin;  begin->prev = newNode;  begin = newNode;  }  }  }  void Deque::insert\_rear(int data)  {  Node\* newNode = Node::getnode(data);  if (newNode == NULL)  cout << "OverFlow\n";  else  {  if (rear == NULL)  begin = rear = newNode;  else  {  newNode->prev = rear;  rear->next = newNode;  rear = newNode;  }  }  }  void Deque::delete\_begin()  {  if ((begin == NULL))  cout << "UnderFlow\n";  else  {  Node\* temp = begin;  begin = begin->next;  if (begin == NULL)  rear = NULL;  else  begin->prev = NULL;  free(temp);  }  }  void Deque::delete\_rear()  {  if ((begin == NULL))  cout << "UnderFlow\n";  else  {  Node\* temp = rear;  rear = rear->prev;  if (rear == NULL)  begin = NULL;  else  rear->next = NULL;  free(temp);  }  }  int Deque::display()  {  if ((begin == NULL))  cout << "Empty" << endl;  else{  cout << begin->data << endl;  cout << rear->data << endl;  }  }  int main(){  int c,i;  Deque d;  do{  cout<<"\n 1.insert at beginning";  cout<<"\n 2.insert at end";  cout<<"\n 3.display";  cout<<"\n 4.deletion from front";  cout<<"\n 5.deletion from rear";  cout<<"\n 6.exit";  cout<<"\n enter your choice:";  cin>>c;  switch(c) {  case 1:  cout<<"enter the element to be inserted";  cin>>i;  d.insert\_begin(i);  break;  case 2:  cout<<"enter the element to be inserted";  cin>>i;  d.insert\_rear(i);  break;  case 3:  d.display();  break;  case 4:  d.delete\_begin();  break;  case 5:  d.delete\_rear();  break;  case 6:  exit(1);  break;  default:  cout<<"invalid choice";  break;  }  } while(c!=7);  } |