**North South University - Spring 2023**

Course: CSE225L Assessment: Lab 9

Section: 06 NSU ID: 2211424642 Name: Joy Kumar Ghosh

// Task-1 Codes

|  |
| --- |
| #include <iostream>  using namespace std;  class FullQueue{  };  class EmptyQueue{  };  template<class ItemType>  class QueType{  public:  QueType();  QueType(int max);  ~QueType();  void MakeEmpty();  bool IsEmpty();  bool IsFull();  void Enqueue(ItemType);  void Dequeue(ItemType&);  private:  int front;  int rear;  ItemType\* items;  int maxQue;  }; |
| template<class ItemType>  QueType<ItemType>::QueType(int max){  maxQue = max + 1;  front = maxQue - 1;  rear = maxQue - 1;  items = new ItemType[maxQue];  }  template<class ItemType>  QueType<ItemType>::QueType(){  maxQue = 501;  front = maxQue - 1;  rear = maxQue - 1;  items = new ItemType[maxQue];  }  template<class ItemType>  QueType<ItemType>::~QueType(){  delete [] items;  }  template<class ItemType>  void QueType<ItemType>::MakeEmpty(){  front = maxQue - 1;  rear = maxQue - 1;  }  template<class ItemType>  bool QueType<ItemType>::IsEmpty(){  return (rear == front);  }  template<class ItemType>  bool QueType<ItemType>::IsFull(){  return ((rear+1)%maxQue == front);  }  template<class ItemType>  void QueType<ItemType>::Enqueue(ItemType newItem){  if(IsFull())  throw FullQueue();  else{  rear = (rear +1) % maxQue;  items[rear] = newItem;  }  }  template<class ItemType>  void QueType<ItemType>::Dequeue(ItemType& item){  if(IsEmpty())  throw EmptyQueue();  else{  front = (front + 1) % maxQue;  item = items[front];  }  } |
| //driver main file  int main()  {  int maxQueue, item, i;  cout << "Enter max queue size: ";  cin >> maxQueue;  cout << endl;  QueType<int> queueList(maxQueue);  //checking queue is empty or not  cout << "Queue is empty or not: ";  if(queueList.IsEmpty()){  cout << "Queue is Empty." << endl << endl;  }  else{  cout << "Queue is not Empty." << endl << endl;  }  //enqueue 4 items  cout << "Enter four item for EnQueue: ";  for(i = 0; i < 4; i++){  cin >> item;  try{  queueList.Enqueue(item);  }  catch(FullQueue){  cout << "Queue is overflow.";  }  }  cout << endl;  //checking queue is empty or not  cout << "Queue is empty or not: ";  if(queueList.IsEmpty()){  cout << "Queue is Empty." << endl << endl;  }  else{  cout << "Queue is not Empty." << endl << endl;  }  //checking queue is full or not  cout << "Queue is empty or not: ";  if(queueList.IsFull()){  cout << "Queue is Full." << endl << endl;  }  else{  cout << "Queue is not Full." << endl << endl;  }  //enqueue another item  cout << "Enter another item for enqueue: ";  cin >> item;  try{  queueList.Enqueue(item);  }  catch(FullQueue){  cout << "Queue is overflow.";  }  cout << endl;  //printing item  cout << "Printing QueueList in inputs order: ";  QueType<int> tempQueue(maxQueue);  for(i = 0; !queueList.IsEmpty(); i++){  queueList.Dequeue(item);  tempQueue.Enqueue(item);  }  for(i = 0; !tempQueue.IsEmpty(); i++){  tempQueue.Dequeue(item);  cout << item << " ";  queueList.Enqueue(item);  }  cout << endl << endl;  //checking queue is full or not  cout << "Queue is empty or not: ";  if(queueList.IsFull()){  cout << "Queue is Full." << endl << endl;  }  else{  cout << "Queue is not Full." << endl << endl;  }  //enqueue another item  cout << "Enter another item for enqueue: ";  cin >> item;  try{  queueList.Enqueue(item);  }  catch(FullQueue){  cout << "Queue is overflow.";  }  cout << endl;  //Dequeue two items  try{  queueList.Dequeue(item);  queueList.Dequeue(item);  }  catch(EmptyQueue){  cout << "Queue Underflow" << endl;  }  //printing item  cout << "Printing QueueList in inputs order: ";  for(i = 0; !queueList.IsEmpty(); i++){  queueList.Dequeue(item);  tempQueue.Enqueue(item);  }  for(i = 0; !tempQueue.IsEmpty(); i++){  tempQueue.Dequeue(item);  cout << item << " ";  queueList.Enqueue(item);  }  cout << endl << endl;  //Dequeue three items  try{  queueList.Dequeue(item);  queueList.Dequeue(item);  queueList.Dequeue(item);  }  catch(EmptyQueue){  cout << "Queue Underflow" << endl;  }  //checking queue is empty or not  cout << "Queue is empty or not: ";  if(queueList.IsEmpty()){  cout << "Queue is Empty." << endl << endl;  }  else{  cout << "Queue is not Empty." << endl << endl;  }  //Dequeue another items  try{  queueList.Dequeue(item);  }  catch(EmptyQueue){  cout << "Queue Underflow" << endl;  }  return 0;  } |
|  |

// Task-2 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-3 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-4 Codes

|  |
| --- |
|  |
|  |
|  |

// Task-5 Codes

|  |
| --- |
|  |
|  |
|  |