**CSE225L – Data Structures and Algorithms Lab**

**Lab 12**

**Stack and Queue using Linked List**

In today’s lab at first we will design and implement the Stack and Queue ADTs using linked list.

|  |  |
| --- | --- |
| **stack.h**  #include "list.h"  template<class T>  class StackLL{  private:  SinglyLinkedList<T> ls;  public:  StackLL();  ~StackLL();  virtual void push(T value);  virtual T pop();  virtual T top();  virtual bool isEmpty();  }; | **stack.cpp**  #include "stack.h"  template<class T> StackLL<T>::StackLL(){}  template<class T> StackLL<T>::~StackLL(){}  template<class T> void StackLL<T>::push(T value)  {  ls.insertAtStart(value);  }  template<class T> T StackLL<T>::pop()  {  T value = ls.getValue(1);  ls.deleteStart();  return value;  }  template<class T> T StackLL<T>::top()  {  T value = ls.getValue(1);  return value;  }  template<class T> bool StackLL<T>::isEmpty()  {  return ls.isEmpty();  } |
| **queue.h**  #include "list.h"  template<class T>  class QueueLL : public SinglyLinkedListWithTail<T>{  private:  SinglyLinkedListWithTail<T> ls;  public:  QueueLL();  ~QueueLL();  virtual void enqueue(T value);  virtual T dequeue();  virtual T frontItem();  virtual bool isEmpty();  }; | **queue.cpp**  #include "queue.h"  template<class T> QueueLL<T>::QueueLL(){}  template<class T> QueueLL<T>::~QueueLL(){}  template<class T> void QueueLL<T>::enqueue(T value)  {  ls.insertAtEnd(value);  }  template<class T> T QueueLL<T>::dequeue()  {  T value = ls.getValue(1);  ls.deleteStart();  return value;  }  template<class T> T QueueLL<T>::frontItem()  {  T value = ls.getValue(1);  return value;  }  template<class T>  bool QueueLL<T>::isEmpty()  {  return ls.isEmpty();  } |

**Task1:**

\* In main function, instantiate an object of stackLL class and another object of QueueLL both of which can contain *char* type values

\* Read a string from user and push its characters (sequentially) into the stack; also enqueue (sequentially) them in the queue. For e.g. if the input string is “hit” then your stack should contain (from top to bottom): ‘t’, ‘i’, ‘h’; and your queue should contain (from front/head to rear/tail): ‘h’, ‘i’, ‘t’

\* Now write appropriate code that use your stack and queue to determine if the input string was a palindrome or not. For e.g. “madam”, “kayak”, “nayan”, etc. are palindromes whereas “hit” is not a palindrome.

**Task2:**

\* In main function, instantiate an object of stackLL class and another object of QueueLL both of which can contain *char* type values

\* Read two strings from user and push the characters in the first string (sequentially) into the stack; also enqueue (sequentially) the characters in the second string in the queue.

\* Now write appropriate code that use your stack and queue to determine if the first string is the reverse of the second string. For e.g. “feat” is the reverse of “taef” but “feet” is not the reverse of “tee”.