Coding Assignment 2: Implement a stack and a queue class with linked-list Assignment

Due: Sun Mar 1, 2026 11:59pmDue: Sun Mar 1, 2026 11:59pm
Ungraded, 100 Possible Points100 Points Possible

Tip: Linked-list coding examples link: Linked lists class w generic coding

Programming Assignment Specifications:

Goal: Program #2 will focus our attention on stacks, queues, and dynamic memory (Linked-lists).

Task: Your job on program #2 is to implement a stack and a queue class from scratch. Keep these classes separate. Then either via a main program or another class, implement the code that will test your classes.

Test Code: Write a simple application to test your stack and queue class methods are working well. Use your queue and stack to manage the input received data from user. All data must be read from user, must be stored in queue or stack prior to processing. Then, retrieve it from the queue or stack to print out the entire queue or stack to prove the retrieving process follow the queue (FIFO) and stack (LIFO)rules.

Data Structures: The stack and/or queue abstractions must be implemented using **linear singly linked lists**. (look at week3_ArrayLinkedLists slides) All memory within the abstractions (nodes, list of **T** elements) must be dynamically allocated. Also, use generic coding format that be able to store **T** type of elements.

Implement the following classes and methods:

- Class SNode with templete T datatype
- Class LinerSinglyLinkedList with the following methods:
  - isEmptyList() - insertElmAtEnd() - removefromFront() - copyList() -deleteList() -nextElm() - addElmAtFront
- Class Stack that include -push() , -pop(), -top(), isEmpty() operations
- Class Queue that include -insert() , -remove(), next(), isEmpty() operations
- Class Test program that includes storeData function, printOut Function
  - storeData: function to read data from user and add it to the queue / stack.
  -printOut: print out element by element from queue / stack on the output screen.

Things you should know...as part of your program: [60% points]

1. **VERY IMPORTANT Implement a stack ADT to perform the push, pop, isEmpty, ... operations as a class. Implement a queue ADT to perform the queue operations as a class.**
2. Use modular design, separating the .h files from the .cpp files is good but you can implement all in one file as well. Remember, .h files should contain the class header and any necessary prototypes.
3. Make sure to define a constructor and destructor for your stack/queue classes. Your destructor must deallocate all dynamically allocated memory if you need them.
4. Each node must be dynamically allocated to be of the appropriate size. Do not use statically allocated arrays in your nodes!
5. Remember to pass by reference whenever possible.

On the due date, turn in:

1) A listing of your C++ program and the output. Your output should show that you have thoroughly tested all boundary conditions of this assignment.[10% points]

Instructions to deliver Programs:

1-Program source files that includes your comments. [10% points]

2- The report pdf file that include your learned notes, your work, and screenshots of your code results must be submitted on the Canvas by due date. [20% points]