CSCI 3320/8325, Program Assignment #2

Goal

The assignment has the following goals:

1. Demonstrate your mastery of recursion
2. Demonstrate your knowledge of how linked lists work
3. Demonstrate your knowledge of stack and queue data structures.

Obtain the Starter Code

You can find the starter code in Python (see the attached file).

This code provides the complete code for a simple list that uses a linked list.

This code also provides some helper functions that you **do not need to edit**.

Take time to familiarize yourself with the 'printInOrder' and 'printNext' functions. Your solution to this assignment will have code very similar to these functions.

Print in Reverse Order (2 points)

Edit the code in 'printInReverseOrder' so that the values are printed in reverse order.

You need to do this using recursion, not by using a loop, copying the list, or modifying the list.

You will probably want to add another function, for example one called 'printReverse' to run the recursion properly. Again, look at 'printInOrder' and 'printNext' if you are stuck.

Reverse the List (2 points)

Edit the code in 'reverseList' so that the elements in the list have been reversed.

You need to do this using recursion, not by using a loop or by copying the list.

You will probably want to add another function, for example one called 'reverseRecursive' to run the recursion properly. Again, look at 'printInOrder' and 'printNext' if you are stuck.

Implement a stack using a Linked List (3 points)

To implement a stack using a singly linked list, all the singly linked list operations should be performed based on Stack operations LIFO (last in first out). Do not add any changes to the implementation of a singly linked list. Use only the available methods.

You must implement the following stack operations:

- push(): Insert a new element into the stack i.e just insert a new element at the beginning of the linked list.

- pop(): Return the top element of the tack i.e simply delete the first element from the linked list.
- top(): Return the top element.
- printStack(): Print all elements in the stack from the top to the bottom.

Note: The complexity of push, pop, and top operations should be O(c).

Implement a queue using a Linked List (3 points)

To implement a queue using a singly linked list, all the singly linked list operations should be performed based on Queue operations FIFO (First in first out). Do not add any changes to the implementation of a singly linked list. Use only the available methods.

You must implement the following stack operations:

- enqueue(): Insert a new element into the queue (i.e just insert a new element at the end of the linked list)
- dequeue(): Return the top element of the queue (i.e simply delete the first element from the linked list)
- front(): Return the front element of the queue
- printQueue(): Print all elements in the queue from the front element.

Note: The complexity of enqueue, dequeue and front operations should be O(c).

Completion

Your program should have a menu driven user interface at the command line with the following options:

>> Enter choice [1-15] from menu below:

1) **Construct a list L1 with a set of elements.**
2) Print the list
3) Print the list in reverse order.
4) Reverse the list
5) **Construct a stack S1 with a set of elements**.
6) Print a top element of S1
7) Pop from S1 (from the front of S1)
8) Push to S1 (to the front of S1)
9) Print S1 (from top to bottom)
10) **Construct a queue Q1 with a set of elements.**
11) Print a front element of Q1
12) Dequeue from the front of Q1
13) Enqueue to the tail of Q1
14) Print Q1 (from front to tail)
15) Exit the program.

For the sub-option ‘1’, ‘5’, and ‘10’, please assume the user provides an initial set of elements in the input string in a comma-separated format (i.e. ‘1, 5, 6, 10’).

Deliverable

A Python source codes (stack-queue-with-singly_linkedlist.py files)
Justification of the complexity of Stack and Queue operations (in PDF or Word):

- push, pop, and top Stack operations. and
- enqueue, dequeue, and front operations.