ECS 32B – Introduction to Data Structures
Homework 03

Due: October 24, 2020, 5:00pm PST

Important:

- Download the hw03.py file from Canvas -> Assignments -> Homework03.
- Write your code (**in Python3**) in the designated positions in hw02.py.
- The purpose of the homework is to practice using the linear data structures we have learned in class so far, i.e., stack, queue, linked list. **Problem 1 must be solved using queue. Problem 2-5 must be solved using linked lists.** The definition for Queue, Node, UnorderedList are included in hw02.py. If you didn't use the corresponding data structures, only half of the points will be given.
- We use the same representation for a linked list as for a Python list. For example, if a linked list contains three nodes with values 1, 2, 3, respectively (where 1 being the value of the head), then we write it as [1,2,3].
- For problem 3-5, do not transform the input linked lists into Python lists and then create a new linked list. You should solve the problems by manipulating the next pointer in nodes, then return the resulting linked list.
- You may upload as many times as you want before the deadline. Each time the autograder will let you know how many of the test cases you have passed/failed. To prevent people from manipulating the autogader, the content of the test cases are hidden.
- Please **do not copy others' answers**. Autograder can detect similar code. Also, for your own benefit, do not try to find solutions online.

Problem 1: Stack2

Implement the stack class (called Stack2) using queue, meaning that the only operations that can be used are the ones defined in the Queue class.

Problem 2: transform(lst)

Given an unordered list (defined using the UnorderedList class above), transform it into a Python list. When the input list is empty, the output Python list should be empty as well.

Example: When the input unordered list is [1,2,3], the output Python list should be [1,2,3] .

Problem 3: concatenate(lst1, lst2)

Given two (possibly empty) unordered lists, concatenate them such that the first list comes first.

Example: When the first input unordered list lst1 is [1, 2, 3] and the second lst2 is [7,8,9], the output unordered list is [1,2,3,7,8,9].

Problem 4: removeNodesFromBeginning(lst, n)

Remove the first n nodes from an (possible empty) unordered list.

Note: $0 \leq n \leq lst.length()$

Example: Given an unordered list [1, 2, 3, 4, 5] and $n = 3$, return the unordered list [4, 5].

Problem 5: removeNodesFromBeginning(lst, i, n)

Starting from the ith node in an (possibly empty) unordered list, remove the next n nodes, not including the ith node.

- We say the head of the input list is the *first* node, i.e., $i = 1$.
- When $i = 0$, your function should start by removing the head.

Note:

- $n \geq 0$
- $i \geq 0$
- $i + n \leq lst.length()$

Example: let lst be an unordered list [1, 2, 3, 4, 5, 6]

1. When $i = 2$ and $n = 3$, return the unordered list [1, 2, 6].
   Explanation: The second node ($i = 2$) is 2. Removing the next three nodes ($n = 3$) means removing nodes 3, 4, and 5. The remaining ones are 1, 2, 6.
2. When $i = 0$ and $n = 3$, return the unordered list [4, 5, 6].
   Explanation: Since $i = 0$, we remove the first three nodes ($n = 3$) from the beginning of lst, which are 1, 2, 3. The remaining ones are 4, 5, 6.