COMP 203 DATA STRUCTURES AND ALGORITHMS
HOMEWORK 2 (Total=100 points)
Deadline: 26.11.2023 23:59

Read the questions and rules carefully. They are clear and well defined.
Rules:
1. No Cheating: You are not allowed to collaborate with your friends and use any kind of websites or AI. If your homework gives a sign of any of them, **directly it will be graded as zero**.
2. Goal: Please do your homework alone. Our main aim is to **learn** whatever we cover so far.
3. Submission: Submit your homework in **a single pdf.** Also, submit your code as **2 (two) .java files. No other file types will be accepted. You will submit only 3 files, 1 .pdf and 2 java files. DON'T USE ZIP/RAR etc. In these cases, your points will be deducted by 30%.**
4. Coding policy: Explain your code in comments. **This is a must!**

QUESTIONS

1. a. Implement a Node<E> class, SinglyLinkedList<E> class and SSLQueue<E> including their constructors in java. **(7x5=35 points)**

b. Implement a function with the name "public Node<E> enqueue(E addedValue)" to add a node with the value *addedValue* in the queue. This function returns the head node of the singly linked list. This will be implemented in SSLQueue<E> class.

Example: This is the queue:
A->B->C->D->null
After enqueue (M) it will be;
A->B->C ->D->M->null returns the head node of the queue.

c. Explain the Big-O complexity of your code for "public Node<E> enqueue (E addedValue)".
d. Write a test case in main to see if your function works. You may choose your data type.
e. Implement a function with the name "public E dequeue()" to dequeue from the queue. This function returns the deleted node value.

Example: This is the queue:
A->B->C->D->null
After dequeue(), it will be;
B->C->null and returns A.

f. Explain the Big-O complexity of your code for "public E dequeue ()".
g. Write a test case in main to see if your function works. You may choose your data type.

Submit this question as **SSLQueue.java along with the pdf.**

2. a. Implement a Node<E> class, SinglyLinkedList<E> class and SSLStack<E> including their constructors in java. **(7x5=35 points)**

b. Implement a function with the name "public Node<E> push(E addedValue)" to add a node with the value *addedValue* in the stack. This function returns the head node of the singly linked list. This will be implemented in SSLStack <E> class.

Example: This is the stack after the operations of push(A), push(B), push(C), push(D).
A->B->C->D->null (top is D here)
After push (M) it will be;
A->B->C->D->M->null returns the head node of the stack.

c. Explain the Big-O complexity of your code for "public Node<E> push(E addedValue)".

d. Write a test case in main to see if your function works. You may choose your data type.

e. Implement a function with the name "public E pop()" to remove element from the stack. This function returns the deleted node value.

Example: This is the stack after the operations of push(A), push(B), push(C), push(D).
A->B->C->D->null (top is D here)
After pop(), it will be;
A->B->C->null and returns D.

f. Explain the Big-O complexity of your code for "public E pop()".

g. Write a test case in main to see if your function works. You may choose your data type.

Submit this question as **SSLStack.java along with the pdf.**