COMP 203 DATA STRUCTURES AND ALGORITHMS

HOMEWORK 1 (Total=100 points)

Deadline: 02.11.2023 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 **a single .java file. No other file types will be accepted. You will submit only 2 files, .pdf and java. No multiple pdf or java files will be accepted. 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 and a SinglyLinkedList<E> class in java. **(7x5=35 points)**

b. Implement a function with the name “public Node<E> addAfter(E v, E addedValue)” to add a node with the value *addedValue* after the node has the value *v* in a singly linked list. This function returns the head of the singly linked list. This will be implemented in SinglyLinkedList class.

Example: This is singly linked list:
A->B->C->D->null
After addAfter(C,M) it will be;
A->B->C->M->D->null returns the head node of the singly linked list.

c. Explain the Big-O complexity of your code for “public Node<E> addAfter(E v,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 deleteAfter(E n)” to delete the node (that comes after the given value *n*) in a singly linked list. This function returns the deleted node value.

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

f. Explain the Big-O complexity of your code for “public E deleteAfter (E n)”.

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

2. a. Implement a Node<E> class and a DoublyLinkedList <E> class in java. **(7x5=35 points)**

b. Implement a function with the name “public Node<E> addAfter(E n, E addedValue)” to add a node with the value *addedValue* after the node that has value *n* in a doubly linked list. This function returns the header node of the doubly linked list. This function will be implemented in DoublyLinkedList class.

Example: This is DLL:
header<=>A<=>B<=>C<=>D<=>trailer
After addAfter(C,M), it will be;
header<=>A<=>B<=>C<=>M<=>D<=>trailer and function returns the header node.

c. Explain the Big-O complexity of your code for “public Node<E> addAfter(E n, 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 deleteAfter(E n)” to delete a node (that comes after the given node that has value *n*) in a doubly linked list. This function returns the deleted node value.

Example: This is DLL:
header<=>A<=>B<=>C<=>D<=>trailer
After deleteAfter(C) it will be;
header<=>A<=>B<=>C<=>trailer and returns D.

f. Explain the Big-O complexity of your code for “public E deleteAfter (E n)”.

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

3. We will use arrays here. **(6x5=30 points)**

a. Implement a function with the name “public E[] addAfter(E[] myArray, E n, E addedValue)” to add a value (*addedValue*) after the value *n* in *myArray*. This function returns the current version of *myArray*.

Example: This is the myArray:
A=1,2,3,4,5,6
After addAfter(A,3,8), it will be;
A= 1,2,3,8,4,5,6 will return A.

b. Explain the Big-O complexity of your code for “public E[] addAfter(E[] myArray, E n, E addedValue)”

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

d. Implement a function with the name “public E[] deleteAfter(E[] myArray, E n)”.
to delete an element (that comes after the value *n*) in *myArray*. This function returns current version of *myArray*.

Example: This is the myArray:
A=1,2,3,4,5,6
After deleteAfter(A,3), it will be;
A= 1,2,3,5,6 and it will return A.

e. Explain the Big-O complexity of your code for “public E[] deleteAfter(E[] myArray, E n)”.

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