

Lab 7: Doubly Linked List with Dummy Head and Tail

I. Requirements

- You are given the file DIntNode.java to use.
- You should NOT add any new instance variables to the existing class file.
- You could use any methods that are already in the given file or that are discussed during the class.

Implement DoublyLinkedListDummy.java with the following detailed requirements:

Q1 (5 pts) This class has two instance variables:

(1) the first instance variable is the head of this doubly linked List, which should be a dummy node (instead of null reference);

(2) the second instance variable is the tail of this doubly linked List, which should be a dummy node (instead of null reference);

You should NOT add any new instance variables.

Q2 (5 pts) Please implement corresponding getter, setter methods for head, tail of this doubly linked list.

```
public DIntNode getHead()
public DIntNode getTail()
public void setHead(DIntNode node)
public void setTail(DIntNode node)
```

Q3 (10 pts) The no-argument constructor which creates the dummy head and tail and link them together.

Q4 (10 pts) A method to add an element from the end of the list.

```
public void addEnd(int element)
```

Please denote (not analyze in detail) the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 8 pts for method implementation; 2 pts for correctly denoting complexity.

Q5 (10 pts) A method to remove the first actual node (i.e., the node that the dummy head points to).

```
public void removeFromHead()
```

Please denote (not analyze in detail) the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 8 pts for method implementation; 2 pts for correctly denoting complexity.

Q6 (10 pts) A method to convert the list to a string.

```
public String toString()
```

This method should return a String with two lines where the first line lists all the nodes starting from the first node, and the second line lists all the nodes starting from the last node. E.g., given the doubly linked list as in slide 3 in lec10_ch4_dlinkedList.pdf, the output should be

(Forward) 12<->28<->12<->34

(Backward) 34<->12<->28<->12

Q7 (10 pts) A method to compute the number of times that the given value e occurs in nodes in this linked list.

```
public int countOccurrence(int e)
```

This method will count the times that e occurs in this doubly linked list. If no node in this list contains data e , return 0. E.g., given the following list:

1<->3<->7<->1

countOccurrence(1) should return 2;

countOccurrence(2) should return 0;

countOccurrence(3) should return 1;

Please denote (not analyze in detail) the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 8 pts for method implementation; 2 pts for correctly denoting complexity.

Q8 (10 pts) A method to remove all the nodes that contain element e from the list.

```
public boolean removeAll(int e)
```

Please denote (not analyze in detail) the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 8 pts for method implementation; 2 pts for correctly denoting complexity.

Q9 (12 pts) A method to truncate this list at position i .

```
public DoublyLinkedListDummy truncate(int i)
```

After executing this method, the size of the list will be i and it should contain only the elements at indices $0, \dots, i-1$. The return value is another DoublyLinkedListDummy that contains the elements at indices $i, \dots, n-1$

Suppose we have a list as below:

1<->2<->3<->4<->5

list.truncate(2) would return a list

3<->4<->5

and list now contains only 1<->2

The precondition is

(1) i is no less than 0

(2) i is no greater than the size of this list

Please denote (not analyze in detail) the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 10 pts for method implementation; 2 pts for correctly denoting complexity.

Q10 (13 points)

```
public void absorb(DoublyLinkedListDummy l2)
```

This method takes as an argument a `DoublyLinkedListDummy`, `l2`, empties it and appends its contents, in order, to the receiver. For example, if `l1` contains 1, 2, 3 and `l2` contains 4, 5, 6, then after calling `l1.absorb(l2)`, `l1` will contain 1, 2, 3, 4, 5, 6 and `l2` will be empty.

Please denote the time complexity of your method in Big-O and put the complexity in the method comment.

Grading note: 10 pts for method implementation; 3 pts for correctly denoting complexity.

Q11 (5 pts) Write test cases to thoroughly test your code.

II. Note

- Specifications for all your classes and methods:

Please properly explain (1) the functionality of the methods, (2) the parameters, (3) the return values, (4) the pre-conditions if there is any;

Please use inline comments, meaningful variable names, indentation, formatting, and whitespace throughout your program to improve its readability.

- You can (but are not required to) design and implement other facilitating methods (E.g., other get and set methods, toString method) to finish the implementation of the required methods.

III. Submission

Submit through canvas a zipped file containing your (1) java file(s) (not .class files).

IV. Grading Criteria

1. The score allocation is beside the questions.
2. Please make sure that you test your code thoroughly by considering all possible test cases. Your code may be tested using more test cases.