CSCI 2110 – Data Structures in Java
Lab exercise LabN3

- Deadline: Friday Jan 29th at 5pm
- Please first read the assignment check list available in the following link. If you do not follow the check list, you lose many points unfortunately. https://web.cs.dal.ca/~nourashr/teaching/cs2110/assignments/assignmentchecklist/
- If you have any question regarding the questions, please send an email to the instructor or TAs.
- Please submit your work on the Moodle portal: https://courses.cs.dal.ca/login/index.php
- You probably need to review lecture slides which are available in the following link. Feel free to use the codes of slides. https://web.cs.dal.ca/~nourashr/teaching/cs2110/lectures/
- We follow Dalhousie's policy of plagiarism. http://www.dal.ca/dept/university_secretariat/academic-integrity.html

In the following question, you can use the implementation of the *DoublyLinkedList* class provided in the slides. Feel free to change it if needed.

Q1) (45 points) Consider a queue-like data structure that supports insertion and deletion at both the front and the rear of the queue. Such an extension of a queue is called **double-ended queue**, or **deque**, which is usually pronounced "deck".

The deque abstract data type is richer than both the stack and the queue ADTs. The interface of deque is defined as:

```
public interface Deque<E>{
    public int size(); // returns the size of the deque
    public boolean isEmpty(); //returns true if the deque is empty
    public E getFirst() throws EmptyDequeException; // returns
    //the first element of the deque; throws exception if it is
    //empty
```

```
public E getLast() throws EmptyDequeException; // returns the
// last element of the deque; throws exception if it is empty

public void addFirst(E element); // inserts an element to be
//the first in the deque

public void addLast(E element); // inserts an element to be
//the last in the deque

public E removeFirst() throws EmptyDequeException; // Removes
//the first element; an exception is thrown if the deque is
//empty

public E removeLast() throws EmptyDequeException; // Removes
//the last element; an exception is thrown if the deque is
//empty

}
```

In this question, you implement the above-mentioned interface based on a generic doubly linked list.

```
public class DlinkDeque<E> implements Deque<E>{…}
```

Run your code and create a table to show a series of twenty operations and their effects on an initially empty deque of Integer objects.

Table 1: a sample table of operations on an initially empty deque

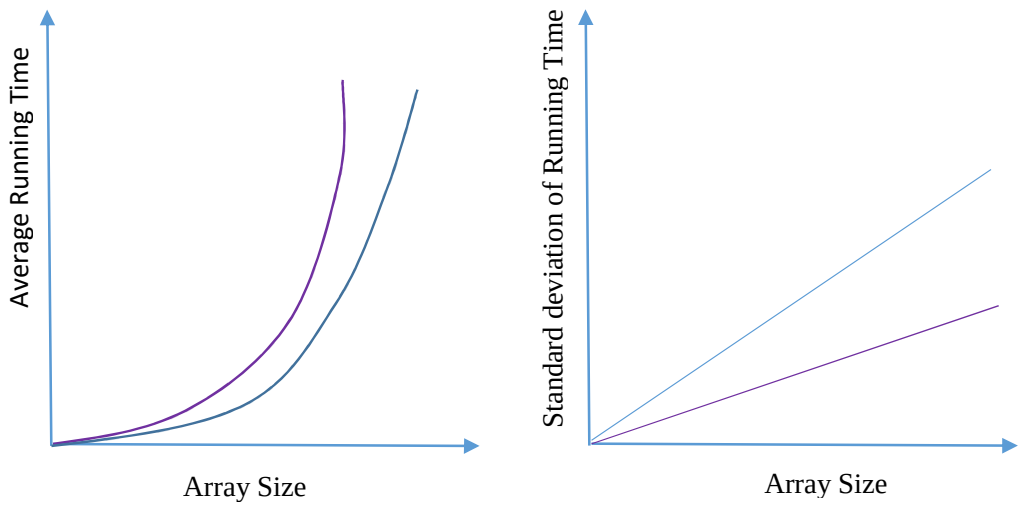
Operation	Output	Deque
addFirst(3)	-	(3)
addFirst(5)	-	(5,3)
removeFirst()	5	(3)

Q2) (40 points)

a) (**10 points**) Implement the insertion sort and the selection sort algorithms to sort an array of int.

b) (**10 points**) Use the theoretical analysis approach and find the running times of these algorithms. For each algorithm, count the number of primitive types, f(n). Express f(n) using Big-oh notation and then prove that your statement is true by finding constants c and n_0. Which algorithm is faster and why?

c) (**20 points**) Use the experimental analysis approach and run your codes on arrays with different sizes. The sizes of the array should be {5000, 10000, 15000, … , 95000, 100000}. **For each size**, run your code five times on randomly-generated arrays and then compute the average and the standard deviations of running times. Plot the average running times of these two algorithms (in ms) vs. the array sizes. Which algorithm is faster? Plot the standard deviations of running times vs. the array sizes. Which algorithm has lower running time deviation on each array size and why?



* Sample plots of average and standard deviation of running times. The curves of your experiments might be different from what are shown here.

*You can use the method `currentTimeMillis()` in `java.lang.System` class to measure running time. To learn more about this class, please read the following link:

(https://docs.oracle.com/javase/7/docs/api/java/lang/System.html)

*You can create random integers using methods provided in `java.util.Random`.

(https://docs.oracle.com/javase/7/docs/api/java/util/Random.html)

Q3) (15 points)

a) (3 points) Show that if f(n) is O(g(n)), then af(n) is O(g(n)), for any constant a > 0.

b) (3 points) Show that if d(n) is O(f(n)) and e(n) is O(g(n)), then the product d(n)e(n) is O(f(n)g(n)).

c) (3 points) Show that if d(n) is O(f(n)) and e(n) is O(g(n)), then d(n) – e(n) is not necessarily O(f(n) – g(n)).

d) (3 points) Show that $(n + 1)^5 + n^4 \log n$ is $O(n^5)$.

e) (3 points) Sort the following ten functions based on running time:

$$4n \log n + 2n \qquad 2^{10} \qquad 2^{\log n}$$

$$4n \qquad 2^n \qquad n^3 \qquad n \log n \qquad n^2 + 10n$$

$$3n + 100 \log n \qquad n^2 \log n$$