CS 284: Homework Assignment 2
Due: October 6, 11:55pm

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. If some code was shown in class, it can be used, but it must be obtained from moodle, the instructor or the TA.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Assignment. Implement a linked list in which a node is linked to its next node as well as to the two previous nodes. Your code must implement the following UML diagrams.

- Your class must be named TwoPrevList.
- TwoPrevList should contain the private inner class shown in the second UML diagram.
- All methods should follow the specifications below.

Hints

- Inserting a node to the beginning or end of the list using the add(int index, Double data) method should be allowed.
- Make sure that all special cases, e.g. an empty list, are handled.

Submit a single file named TwoPrevList.java. No report is required. Your grade will be determined as follows:

- You will get 0 if your code does not compile.
- The code must implement the following UML diagram precisely. (You can add data members and methods as needed.) It will be tested using a driver.
- The code must include tests for all methods implemented.
- We will NOT try to feed erroneous and inconsistent inputs to your methods this time.

TwoPrevList
- head : TPNode - tail: TPNode - size : int

+ TwoPrevList () // Creates an empty list
+ add (in i : int, in d : Double) : boolean // Adds d at index i
+ addFirst (in d : Double) : boolean // Adds d as the new head
+ addLast (in d : Double) : boolean // Adds d as the new tail
+ get (in i : int) : Double // Returns the number stored at index i
+ size() : int // Returns the list size
+ removeFirst () : Double // Removes and returns the data at the head
+ removeLast () : Double // Removes and returns the data at the tail
+ remove (in i : int) : Double // Removes and returns the element at index i
+ find (in d : Double) : int // Returns the index of the first
// instance of d, or -1 if d is not in the list
+ average(in i : int): Double // Returns the average of the number at index i
// and the previous two numbers on the list. If only one or no previous number
// exists, returns the average of the available data
+ toString() : void // returns a String with all the elements in the list
//separated by commas

TPNode
+ data : Double + next : TPNode + prev : TPNode + prev2 : TPNode

+ TPNode (in d : Double) // Creates a node holding d
+ TPNode (in d : Double, in n : TPNode, in pr : TPNode, in pr2 : TPNode)
// Creates a node holding d, with n as next,
//pr as prev and pr2 as prev2