School of Computer Science

COMP20010: Data Structures and Algorithms I
Semester I

Practical 3
On: 31/10/2017
Due: 08/11/2017

Q1: _____ **(10points)**

Start with the outline of the class in `src/SinglyLinkedList.java`.

Referring to the lecture notes and the book (Goodrich et al), fill in the bodies of the missing functions to implement a working Singly Linked List class which implements the List ADT.

Q2: _____ **(10points)**

The starter code in the `SinglyLinkedList` class is non-generic- it is designed to hold only `String`'s.

Write a generic version of the Singly Linked List class.

Q3: _____ **(10points)**

Referring to the lecture notes and the book (Goodrich et al), implement a Doubly Linked List class.

Q4: _____ **(10points)**

Once you have the Singly Linked List and Doubly Linked List class working, then test them with the code in `src/LinkedListTester`.

```
1  public class LinkedListTester {
2          private static Random random = new Random(20010);
3
4          public static void test1() {
5                  // create your own LinkedList
6                  SinglyLinkedList ll = new SinglyLinkedList();
7
8                  // lets create an array of String's
9                  // and fill our list with a random sample of the data
10                 String[] data = { "one", "two", "three", "four", "five"↩
                       , "six", "seven", "eight", "nine", "ten" };
11
12                 for (int i = 0; i < 50; ++i) {
13                         ll.addLast(data[random.nextInt(data.length)]);
14                 }
15
16                 // now, call each function in the API, choosing at ↩
                       random
17                 int N = 100;
```

```
18                  String[] procs = { "addFirst", "addLast", "removeFirst"↩
                        , "removeLast", "addBefore", "remove" };
19
20                  for (int i = 0; i < N; ++i) {
21                          String s = data[random.nextInt(data.length)];
22                          switch (procs[random.nextInt(procs.length)]) {
23                          case "addFirst":
24                                  ll.addFirst(s);
25                                  break;
26                          case "addLast":
27                                  ll.addLast(s);
28                                  break;
29                          case "removeFirst":
30                                  if (!ll.isEmpty()) {
31                                          ll.removeFirst();
32                                  }
33                                  break;
34                          case "removeLast":
35                                  if (!ll.isEmpty()) {
36                                          ll.removeLast();
37                                  }
38                                  break;
39                          case "remove":
40                                  ll.remove(s);
41                                  break;
42                          case "addBefore":
43                                  // if you have a positional add() then ↩
                                        do this:
44                                  // ll.add(random.nextInt(ll.size()), s)↩
                                        ;
45
46                                  // if you have a key-based add do this
47                                  // String key = data[random.nextInt(↩
                                        data.length)];
48                                  // ll.addBefore(key, s);
49                                  break;
50                          default:
51                                  System.out.println("unknown");
52                                  break;
53                          }
54                  }
55
56                  // print out the size of the list and the elements...
57                  System.out.println("size(ll): " + ll.size());
58                  for (String s : ll) {
59                          System.out.println("ll -> " + s);
60                  }
61          }
62
63          public static void main(String[] args) {
64                  test1();
65          }
```

```
66 }
```

Q5: _____ **(10points)**

Describe a method which finds the middle node of a Doubly Linked List.

Implement and test this method on your own implementation of the DLL.

Q6: _____ **(10points)**

Reimplement your `Scoreboard` class from Tutorial 1 using a DoublyLinkedList, instead of an array.