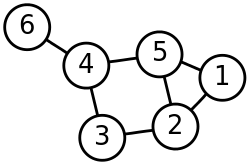
Lab 3

Linked LIsts

1. Nodes. (5 points, 0.5 points per question)

A node has a value, and links to other nodes. The links may or may not have arrows, depending on what type of system is being represented.



Thanks to <https://en.wikipedia.org/wiki/Node_(computer_science)>

1.1. How many nodes are in this diagram?

**- 6**

1.2. How many links to other nodes (“edges”) are in this diagram?

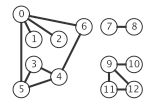
**- 7**

1.3. What three nodes is node 4 directly connected to?

**- 3, 5, and 6.**

1.4. Can you reach node 6 from node 1?

**- Yes, 1 -> 5 -> 4 -> 6.**

  
Thanks to <http://algs4.cs.princeton.edu/41graph/>

1.5. How many nodes are in this diagram?

**-13**

1.6. How many links to other nodes (“edges”) are in this diagram?

**-13**

1.7. Can you reach node 6 from node 1?

**-Yes, 1 -> 0 -> 6.**

1.8. Can you reach node 9 from node 2? Why not?

**-No, they are not linked.**

1.9. Does every node on a graph have to be connected to every other node?

**-No**

1.10. Which node connects to more of the graph: node 4 or node 8?

**-Node 4.**

2.Singly-linked lists. (10 points)

The textbook’s code for a singly-linked list class is included with the lab files. Use this to create an Integer Integer singly-linked list:

5 -> 2 -> 3 -> 8 -> 1 -> 7.

Output the values from the singly-linked list to the console.

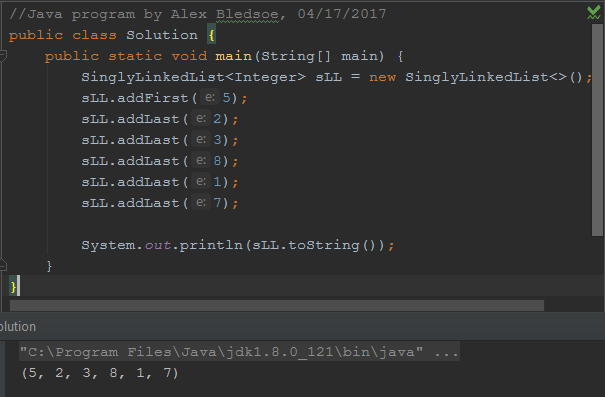
Sample program output:



I am not picky about output formatting, but the values displayed must come from the nodes of the singly-linked list, and not from the main program.

Rubric:  
Student name and today’s date is a comment in the first line of the programs: -5 points if fails  
Instantiate an Integer singly-linked list: 2 points  
Create list with the required values, in order: 4 points  
Output values from list data structure: 4 points

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your main program's .java file, here. You do not need to include the code from the textbook's SinglyLinkedLists.java file.



**Solution.java:**

//Java program by Alex Bledsoe, 04/17/2017  
public class Solution {  
 public static void main(String[] main) {  
 SinglyLinkedList<Integer> sLL = new SinglyLinkedList<>();  
 sLL.addFirst(5);  
 sLL.addLast(2);  
 sLL.addLast(3);  
 sLL.addLast(8);  
 sLL.addLast(1);  
 sLL.addLast(7);  
  
 System.*out*.println(sLL.toString());  
 }  
}

3. Doubly-linked lists (15 points)

3.1. The textbook’s code for a doubly-linked list class is included with the lab files. Use this to create an Integer doubly-linked list DLL1:

5 <-> 2 <-> 3 <-> 8 <-> 1 <-> 7 <-> 2 <-> 7 <-> 9 <-> 2 <-> 1.

3.2. Output all the values from the DLL1 to the console.

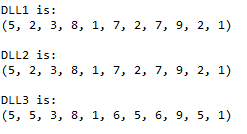
3.3. Using DLL1.removeFirst(), make a copy of the list to a new doubly-linked list, DLL2.

3.4. Output all the values from DLL2 to the console.

3.5. Using DLL2.removeLast(), make a copy of the list to a new doubly-linked list, DLL3.  
3.5.1. While copying, change all nodes with value 2 to value 5. You should not insert any nodes with value 2 into DLL3.  
3.5.2 While copying, change all nodes with value 7 to value 6. You should not insert any nodes with value 7 into DLL3.

3.6. Output all the values from DLL3 to the console.

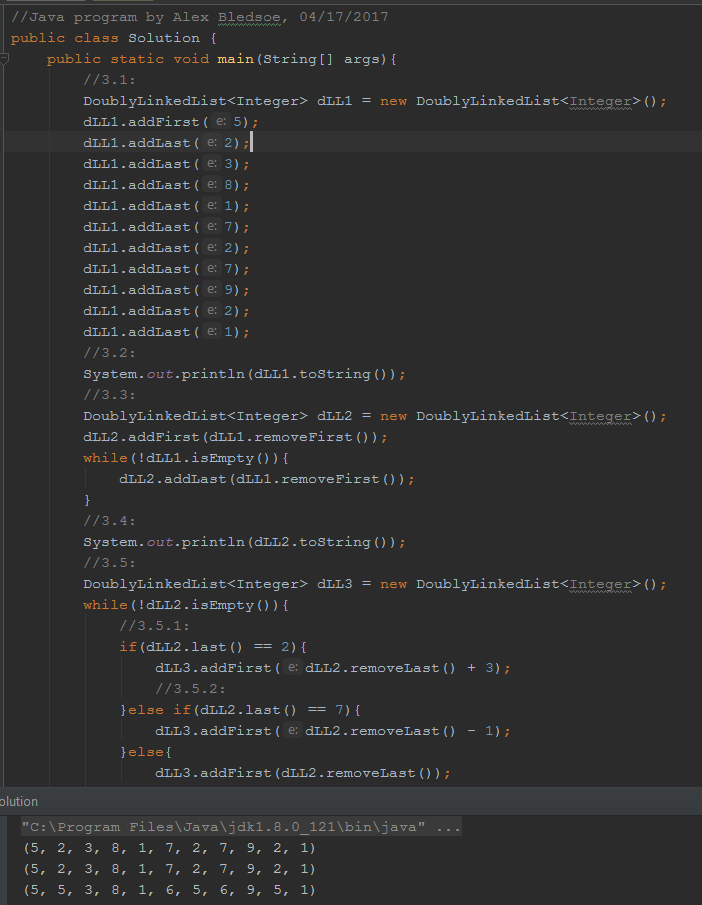
Sample program output:



I am not picky about output formatting, but the values displayed must come from the nodes of the doubly-linked list, and not from the main program.

Rubric:  
Student name and date is in a comment on the first line of the programs: -5 points if fails  
Create DLL1 with correct values: 2 points  
Create DLL2 using DLL1.removeFirst: 4 points  
Create DLL3 using DLL2.removeLast: 4 points  
Change node values before adding to DLL3: 3 points  
Output from DLL1, DLL2, DLL3: 2 points

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your main program's .java file, here. You do not need to include the code from the textbook's DoublyLinkedLists.java file.



**Solution.java:**

//Java program by Alex Bledsoe, 04/17/2017  
public class Solution {  
 public static void main(String[] args) {  
 //3.1:  
 DoublyLinkedList<Integer> dLL1 = new DoublyLinkedList<Integer>();  
 dLL1.addFirst(5);  
 dLL1.addLast(2);  
 dLL1.addLast(3);  
 dLL1.addLast(8);  
 dLL1.addLast(1);  
 dLL1.addLast(7);  
 dLL1.addLast(2);  
 dLL1.addLast(7);  
 dLL1.addLast(9);  
 dLL1.addLast(2);  
 dLL1.addLast(1);  
 //3.2:  
 System.*out*.println(dLL1.toString());  
 //3.3:  
 DoublyLinkedList<Integer> dLL2 = new DoublyLinkedList<Integer>();  
 dLL2.addFirst(dLL1.removeFirst());  
 while (!dLL1.isEmpty()) {  
 dLL2.addLast(dLL1.removeFirst());  
 }  
 //3.4:  
 System.*out*.println(dLL2.toString());  
 //3.5:  
 DoublyLinkedList<Integer> dLL3 = new DoublyLinkedList<Integer>();  
 while (!dLL2.isEmpty()) {  
 //3.5.1:  
 if (dLL2.last() == 2) {  
 dLL3.addFirst(dLL2.removeLast() + 3);  
 //3.5.2:  
 } else if (dLL2.last() == 7) {  
 dLL3.addFirst(dLL2.removeLast() - 1);  
 } else {  
 dLL3.addFirst(dLL2.removeLast());  
 }  
 }  
 //3.6:  
 System.*out*.println(dLL3.toString());  
 }  
}

4.Comprehensive. (15 points)

Thanks to Goodrich 6th Edition, reinforcement exercise R-3.8

Modify the textbook’s double-linked list class. Add a method that finds the middle node of a doubly linked list by link hopping, without relying on explicit knowledge of the size of the list. In the case of an even number of nodes, report the node slightly left of center as the middle. You can assume the list has at least one item.

Your findMiddle() code will be located in DoublyLinkedList.java. **If you’re using size(), you’re doing it wrong.**

Starter code for the findMiddle() method:

**// Janet McGregor 4/20/2053**

**public** E findMiddle() {

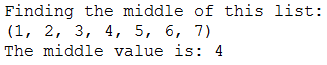
Node<E> middleNode = header.next;

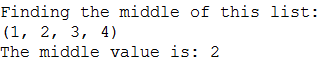
Node<E> currentNode = header.next;

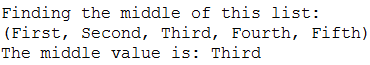
// the rest of your code goes here  
**return** middleNode.element;

}

Sample program output is below:







**Your code must work for any doubly-linked list, not just the examples.**

Rubric:

Student name and today’s date are a comment on the first line of the program (-5 if fails)

findMiddle works without referencing the size property of the list: (-10 if fails)  
Program output shows three test cases with different data types, regardless of whether findMiddle() works or not: 5 points

findMiddle works for lists with an odd number of items: 5 points

findMiddle works for lists with an even number of items: 5 points

Please paste a screenshot of a successful program run, and copy-and-paste the source code for **findMiddle()** from DoublyLinkedList.java .



**Solution.java (demonstration class):**

//Java program by Alex Bledsoe, 04/19/2017  
public class Solution {  
 public static void main(String[] args) {  
 //Creates 3 doubly linked lists to test.  
 DoublyLinkedList<Integer> dLL1 = new DoublyLinkedList<>();  
 DoublyLinkedList<Integer> dLL2 = new DoublyLinkedList<>();  
 DoublyLinkedList<String> dLL3 = new DoublyLinkedList<>();  
  
 //Assigns elements to each node in each list.  
 dLL1.addFirst(1);  
 dLL1.addLast(2);  
 dLL1.addLast(3);  
 dLL1.addLast(4);  
 dLL1.addLast(5);  
  
 dLL2.addFirst(1);  
 dLL2.addLast(2);  
 dLL2.addLast(3);  
 dLL2.addLast(4);  
  
 dLL3.addFirst("First");  
 dLL3.addLast("Second");  
 dLL3.addLast("Third");  
 dLL3.addLast("Fourth");  
 dLL3.addLast("Fifth");  
 dLL3.addLast("Sixth");  
  
 //Calls findMiddle() method for each list and prints to the console.  
 System.*out*.println(dLL1.toString());  
 System.*out*.println("The middle value is " + dLL1.findMiddle() + ".\r\n");  
 System.*out*.println(dLL2.toString());  
 System.*out*.println("The middle value is " + dLL2.findMiddle() + ".\r\n");  
 System.*out*.println(dLL3.toString());  
 System.*out*.println("The middle value is " + dLL3.findMiddle() + ".\r\n");  
 }  
}

**findMiddle() method added to DoublyLinkedList.java:**

E findMiddle() {  
 Node<E> middleNode = header.next;  
 Node<E> currentNode = header.next;  
  
 while ((currentNode.next).next != trailer & (currentNode.next).next != null) {  
 currentNode = (currentNode.next).next;  
 middleNode = middleNode.next;  
 }  
 return middleNode.element;  
}