Homework 2

Seungwook Han
sh3264
Data Structures
Professor Blaer

1. Problem 1

```
public static void printLots(ArrayList<Integer> L,
ArrayList<Integer> P) {
	Iterator<Integer> itrL = L.iterator();
	Iterator<Integer> itrP = P.iterator();
	int count = -1;
	int element = 1000;
	int position;

	if (P.isEmpty()) {
		System.out.println("Position List is empty!");
	}

	if (L.isEmpty()) {
		System.out.println("Element List is empty!");
	}

	while (itrP.hasNext()) {
		position = itrP.next();
		while (count != position) {
			count++;
			if (itrL.hasNext()) {
				element = itrL.next();
			} else {
				System.out.println("Error: Out of Bound");
				break;
			}
		}
		if (itrL.hasNext()) {
			System.out.println("The integer in the position
			is: " + element);

		}

	}

}
```

2. Problem 2 : Weiss 3.4

    a. We have two sorted Linked Lists L1 and L2
    b. Create another list called Result
    c. Set the nodes of L1 and L2 to the head
    d. While (L1_node and L2_node != null), run code below
        i. If L1_node and L2_node values are equal to each other, add the value to the Result list
        ii. If L1_node's value is smaller than L2_node's value, set L1_node to L1_node.next (its next node)
        iii. If L1_node's value is greater than L2_node's value, set L2_node to L2_node.next (its next node)

3. Problem 3 : Weiss 3.24

    a. Create an array called twoStackArray: int[] twoStackArray = new int[n]
    b. Create two int variables called s1Index and s2Index
        i. private int s1Index;
        ii. private int s2Index;
    c. Set s1Index to -1
        i. s1Index = -1;
    d. Set s2Index to the length of twoStackArray
        i. s2Index = twoStackArray.length()
    e. Push1 - Push Method for s1 / Stack 1, int pushInt
        i. If (s1Index == s2Index)
            System.out.println("Stack Overflow: Every slot in the array is used")
        ii. Else,
            1. Increment the respective s1Index by 1 and set twoStackArray[s1Index] – the element in the position of s1Index in the array – to the integer being pushed
            2. s1Index ++;
            3. twoStackArray[s1Index] = pushInt;

    f. Push2 – Push Method for s2 / Stack 2, int pushInt
        i. If s1Index == s2Index
            System.out.println("Stack Overflow: Every slot in the array is used")
        ii. Else,
            1. Decrement the respective s2Index by 1 and set twoStackArray[s2Index] – the element in the position of s2Index in the array – to the integer being pushed
            2. s2Index--;
            3. twoStackArray[s2Index] = pushInt;

g. Pop1 - Pop Method for s1 / Stack 1
    i. If s1Index != -1,
        1. Then set an int variable popInt – the integer that will be popped – to twoStackArray[s1Index] – the element in the position of s1Index in the array
            a. int popInt = twoStackArray[s1Index]
        2. Then decrement s1Index by 1
            a. s1Index --;
        3. Then return popInt
            a. return popInt;
    ii. Else
    System.out.println("Stack 1 is empty!")
    return -1;

h. Pop2 – Pop Method for s2 / Stack 2
    i. If s2Index != twoStackArray.length
        1. Then set an int variable popInt – the integer that will be popped – to twoStackArray[s2Index] – the element in the position of s2Index in the array
            a. int popInt = twoStackArray[s2Index]
        2. Then increment s2Index by 1
            a. s2Index++;
        3. Then return popInt
            a. return popInt
    ii. Else
    System.out.println("Stack 2 is empty!");
    return -1;

4. Problem 4 – MTA Problem
    a. Solution for this specific input train and 3 holding tracks as a sequence of steps
        i. Move car #4 from the front of the input track to the top of the holding track S1
        ii. Move car #3 from the front of the input track to the top of the holding track S1
        iii. Move car #1 from the front of the input track to the back of the output track
        iv. Move car #8 from the front of the input track to the top of the holding track S2
        v. Move car #2 from the front of the input track to the back of the output track
        vi. Move car #3 from the top of the holding stack S1 to the back of the output track
        vii. Move car #4 from the top of the holding stack S1 to the back of the output track
        viii. Move car #7 from the front of the input track to the top of the holding track S2

ix. Move car #6 from the front of the input track to the top of the holding track S2
x. Move car #9 from the front of the input track to the top of the holding track S3
xi. Move car #5 from the front of the input track to the back of the output track
xii. Move car #6 from the top of the holding track S2 to the back of the output track
xiii. Move car #7 from the top of the holding track S2 to the back of the output track
xiv. Move car #8 from the top of the holding track S2 to the back of the output track
xv. Move car #9 from the top of the holding track S3 to the back of the output track

b. Show an example for a train length 9 that cannot be rearranged in increasing order using 3 holding tracks
    i. [2,5,1,7,6,4,3,9,8]