Data Structure HW2

1. You are given a list, L, and an another list, P, containing integers sorted in ascending order. The operation printLots(L,P) will print the elements in L that are in positions specified by P. For instance, if P=1,3,4,6, the elements in positions 1,3,4, and 6 in L are printed. Write the procedure printLots(L,P). The code you provide should be the java method itself (not pseudocode), the containing class is not necessary. You may use only the public Collection (Links to an external site.) methods that are inherited by lists L and P. You may not use methods that are only in List. Anything that the Collection methods return is fair game, so you might think about how to use iterators.

---

code as follow

```
public void printLots(List<Integer> L, List<Integer> P) {

        Iterator<Integer> itrP = P.iterator();
        Integer lengthL = new Integer(L.size());

        while (itrP.hasNext()) {
            Integer i = itrP.next();
            if (lengthL.compareTo(i) > 0) {
                System.out.println(L[i]);
            }
            else {
                System.out.println("List L is read through");
                break; //leave the while loop
            }
        }
    }
```

2. Weiss 3.4 - provide java-like pseudocode for this problem

```
    public List<T> intersection2(List<T> L1, List<T> L2) {
        List<T> L3 = new ArrayList<T>();
        int i = 0;
        int j = 0;

        while ((i<L1.size()) && (j<L2.size())) {
            if (L1[i] == L2[j]) {
                if (!L3.contains(L1[i])) {
```

```
                L3.add(L1[i]);
            }
            i++;
            j++;
        }
        else if (L1[i] < L2[j]) {
            i++;
        }
        else {
            j++;
        }
    }
    return L3;
}
```

3. Weiss 3.24 - provide java-like pseudocode for this problem

---

```
public class TwoStacksInAnArray {
    int[] array;
    int headOne, headTwo;

    public TwoStacksInAnArray(int n) {
        array = new int[n];
        headOne = -1;
        headTwo = array.length;
    }

    public void pushOne(int data) {
        if ( (headTwo-headOne) > 1) {
            array[++headOne] = data;
        }
        else {
            System.out.println("There is no space in stack1");
        }
    }

    public void pushTwo(int data) {
        if ((headTwo-headOne) > 1) {
```

```
            array[--headTwo] = data;
        }
        else {
            System.out.println("There is no space in stack2");
        }
    }

    public int popOne() {
        if (headOne > -1) {
            return array[headOne--];
        }
        else {
            System.out.println("There is no element in stack1");
            return 0;
        }
    }

    public int popTwo() {
        if (headTwo < array.length) {
            return array[headTwo++];
        }
        else {
            System.out.println("There is no element in stack2");
            return 0;
        }
    }

    public boolean isEmptyOne() {
        return (headOne == -1);
    }

    public boolean isEmptyTwo() {
        return (headTwo == array.length);
    }

    public int sizeOne() {
        return (headOne + 1);
    }

    public int sizeTwo() {
        return (array.length - headTwo);
```

```
    }
}
```

4. In the MTA Subway system, occasionally cars on a train need to be re-arranged. For instance, assume we label the cars of a train with the number `[5,9,6,7,2,8,1,3,4]` (the right end of the list is the front of the train - in this case 4), and we would like to arrange the cars like this: `[9,8,7,6,5,4,3,2,1]`

   (a) Provide a solution for this specific input train and 3 holding tracks as a sequence of steps.

   (b) Show an example for a train of length 9 that cannot be rearranged in increasing order using 3 holding tracks.

(a)
4 →S1
3 →S1
1 →Output //[1]
8 →S2
2 →Output //[2,1]
3 →Output //[3,2,1]
4 →Output //[4,3,2,1]
7 →S2
6 →S2
9 →S3
5 →Output //[5,4,3,2,1]
6 →Output //[6,5,4,3,2,1]
7 →Output //[7,6,5,4,3,2,1]
8 →Output //[8,7,6,5,4,3,2,1]
9 →Output //[9,8,7,6,5,4,3,2,1]

(b) [1,9,8,7,6,5,4,3,2]