COMSW3134
Homework 2

Jennifer Bi (jb3495)

October 7, 2016

1. Problem 1

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

    Iterator<T> itL = new L.iterator();
    Iterator<Integer> itP = new P.iterator();
    int count;
    int nextItem;

    while (itP.hasNext() && itL.hasNext()){
        nextItem = itP.next();
        count = 0;
        while (count<nextItem){
            itL.next();
            count++;
        }
        System.out.println(itL.next());
    }
}
```

2. Weiss 3.4

Given two sorted lists, L_1 and L_2, write a procedure to compute $L_1 \cap L_2$ using only the basic list operations.

```
public intersect(L1, L2){
    Iterator iteratorL_1 = new iterator
    Iterator iteratorL_2 = new iterator

    while (count < (L1.size, L2.size)) and
        (iteratorL2.hasNext and iteratorL1.hasNext) {
        count++
```

```
        elt = L1.next
        if elt = L2.next
            print elt
    }


}
```

3. Weiss 3.24

Write routines to implement two stacks using only one array. Your stack routines should not declare an overflow unless every slot in the array is used.

```
public class TwoArrayStack

    Object[] array

    public MyStack(){
    array = new Object[n];
    size1 = 0;
    size2 = 0;
    }

    public void push1(element){
        if (size1 < n-size2-1)
            Object[size1] = element
            size1++;

        else declare overflow
    }

    public void push2(element){
        if (size1 < n-size2-1)
            Object[n-size2] = element
            size2++;

        else declare overflow
    }

    public T pop1(){
        --size1;
        return Object[size1+1];
        // add back 1 since we subtracted from size1 before returning

    }
```

```java
public T pop2(){
    --size2;
    return Object[n-size2-1];
    // subtract out 1 since we subtracted from size1 before returning

}

public int top1(){
    return size1;

public int top2(){
    return size2;
}
```

4. Problem 4

a) To reorganize the trains from positions [5,9,6,7,2,8,1,3,4] to [9,8,7,6,5,4,3,2,1], we proceed in the following steps:

1. Car 4, input $\rightarrow S_1$
2. Car 3, input $\rightarrow S_1$
3. Car 1, input $\rightarrow$ output
4. Car 8, input $\rightarrow S_2$
5. Car 2, input $\rightarrow$ output
6. Car 7, input $\rightarrow S_2$
7. Car 6, input $\rightarrow S_2$
8. Car 9, input $\rightarrow S_3$
9. Car 5, input $\rightarrow S_2$, thus no cars remain in the input
10. Car 3, $\mathrm{S}_1 \rightarrow$ output
11. Car 5, 6, 7, 8, $\mathrm{S}_2 \rightarrow$ output
12. Car 9, $\mathrm{S}_3 \rightarrow$ output, thus all cars are in output in the desired order.

b) Consider the train with cars ordered [7,5,3,1,8,6,4,2,9]. We will show that there is no possible sequence of steps to rearrange the cars in increasing order. If such an arrangement is possible, it will be accomplished through two instructions: For car i in the input track, *1)* if $i < x$ for all cars x in the output, move car i to output and *2)* If not, move car i to a holding cell such that $i < x$ for all cars x in the holding track. (Rule *2)* is in consideration of the first-in first-out fashion of the holding tracks, so that we may eventually move cars from holding tracks to output and guarantee increasing order.) Thus, we move from the input car 9 $\rightarrow S_1$, car 2 $\rightarrow S_2$, car 4 $\rightarrow S_1$, car 6 $\rightarrow S_1$ since $6 > 1$ and $6 > 2$ but $6 < 9$. We find that following 1) and 2) does not allow for car 8 to be placed in any track. Thus, the arrangement is not possible.