Homework 2

Matthew Lerner-Brecher

October 10, 2016

Problem 1 :

```
public static void printLots(List L, List<Integer> P) {
   int i = 0;
   Integer pval = 0;
   Iterator piter = P.iterator();
   Iterator liter = L.iterator();
   while (piter.hasNext()) {
      pval = (Integer) piter.next();
      while (i < pval) {
         if (liter.hasNext()) {
            liter.next();
         }
         i += 1;
      }
      if (liter.hasNext()) {
         System.out.println(liter.next());
      }
      else {
         System.out.println("No element at " + pval);
      }
      i += 1;
   }
}
```

Problem 2 :

```
List intersect(List A, List B) {
   List newList = new List();
   aindex = 0
   bindex = 0
   while (aindex < A.size() && bindex < B.size()) {
      if (A.get(aindex).compareTo(B.get(bindex)) < 0) {
         aindex += 1
      }
```

```
      else if (A.get(aindex).compareTo(B.get(bindex)) > 0) {
         bindex += 1;
      }
      else {
         newList.add(A.get(aindex));
      }
   }
   return newList;
}
```

Problem 3 :

```
AnyType[] a;
int indexA;
int indexB;

Stack(int i) {
   a = new AnyType[i];
   indexA = 0;
   indexB = i-1;
}

void pushA(AnyType x) {
   if (indexA > indexB) {
      throw StackOverFlowException;
   }
   else {
      a[indexA] = x;
      indexA += 1;
   }
}

void pushB(AnyType x) {
   if (indexA > indexB) {
      throw StackOverFlowException;
   }
   else {
      a[indexB] = x;
      indexB -= 1;
   }
}

AnyType popA() {
   AnyType temp = a[indexA - 1];
   a[indexA - 1] = 0;
   indexA -= 1;
   return temp;
}

AnyType popB() {
   AnyType temp = a[indexB + 1];
```

```
    a[indexB + 1] = 0;
    indexB += 1;
    return temp;
}
```

Problem 4a : The following sequence of moves will arrange it as desired:

- Move Car 4 from input to S_1
- Move Car 3 from input to S_1
- Move Car 1 from input to output
- Move Car 8 from input to S_2
- Move Car 2 from input to output
- Move Car 3 from S_1 to output
- Move Car 4 from S_1 to output
- Move Car 7 from input to S_2
- Move Car 6 from input to S_2
- Move Car 9 from input to S_1
- Move Car 5 from input to output
- Move Car 6 from S_2 to output
- Move Car 7 from S_2 to output
- Move Car 8 from S_2 to output
- Move Car 9 from S_1 to output

Problem 4b : We first note the following: suppose we have two cars x_1 and x_2 in the same holding track with $x_1 > x_2$ and x_1 closer to the top of the track than x_2. In order to get these to the output track they need to be popped off of the holding track. However, x_1 must be popped off before x_2 due to how a stack works. However, if x_1 is popped off of the stack before x_2, then it will end up closer to the front of the output track than x_2. This means the output track is not ordered as desired as $x_1 > x_2$. Hence we want to avoid the aforementioned arrangement.
We now claim $[1, 9, 8, 7, 6, 5, 4, 3, 2]$ is impossible to be rearranged in increasing order. Since we want 1 at the front of the output, we need to push the first 8 elements into one of the holding tracks first. We first push 2 into an arbitrary holding track (say S_1). By our above

reasoning, we need to push 3 into a different holding track (say S_2). Once again by the above reasoning, we need to push 4 into the last holding track S_3. However, by the above reasoning, 5 must be in a different holding track from 2,3,4, but all the holding tracks are full so this is impossible. Hence we can't rearrange $[1, 9, 8, 7, 6, 5, 4, 3, 2]$ in increasing order.