

written.pdf
Data Structures – Homework 2
Connor Goggins

1.

```
public void printLots(List<AnyType> L, List<Integer> P) {
    java.util.Iterator<Integer> p = new P.iterator();

    while(p.hasNext()) {
        int pElement = p.next();
        java.util.Iterator<AnyType> l = new L.iterator();
        int index = 0;

        boolean match = false;

        if(pElement < L.size() && pElement > -1) {
            while(l.hasNext() && match == false) {
                if (pElement == index){
                    System.out.println(l.next());
                    match = true;
                }
                else {
                    index++;
                    l.next();
                }
            }
        }
    }
}
```

2.

```
public <AnyType extends Comparable<AnyType>> List<AnyType>
intersect(List<AnyType> L1, List<AnyType> L2) {
    List<AnyType> intersection = new List<AnyType>();

    java.util.Iterator<AnyType> iterator1 = new
    L1.iterator();
    java.util.Iterator<AnyType> iterator2 = new
    L2.iterator();

    while(iterator1.hasNext()) {
        AnyType tempVar1 = iterator1.next();
        while(iterator2.hasNext()) {
            AnyType tempVar2 = iterator2.next();
            if (tempVar1.compareTo(tempVar2) == 0) {
                intersection.add(tempVar1);
            }
        }
    }
}
```

```

        }
    }

    return intersection;
}

```

3.

```

AnyType[] arr = new AnyType[n]; // n is the size of the
array

```

```

int currentPushIndex1 = -1;
int currentPushIndex2 = n;

```

```

stack1.push(AnyType k) {
    currentPushIndex1++;
    if(currentPushIndex1 > currentPushIndex2)
        throw overflowError;
    else
        arr[currentPushIndex1] = k;
}

```

```

stack1.pop() {
    if(currentPushIndex1 < 0)
        throw underflowError;
    else
        arr[currentPushIndex1] = k;
        currentPushIndex1--;
        return k;
}

```

```

stack2.push(AnyType k) {
    currentPushIndex1--;
    if(currentPushIndex2 < currentPushIndex1)
        throw overflowError;
    else
        arr[currentPushIndex2] = k;
}

```

```

stack2.pop() {
    if(currentPushIndex2 > n - 1)
        throw underflowError;
    else
        arr[currentPushIndex2] = k;
        currentPushIndex2++;
        return k;
}

```

4.

a.

- i. Car 4 from input track to S1
- ii. Car 3 from input track to S1
Car 1 from input track to back of output track
- iii. Car 8 from input track to S2
- iv. Car 2 from input track to back of output track
- v. Car 3 from S1 to back of output track
- vi. Car 4 from S1 to back of output track
- vii. Car 7 from input track to S2
- viii. Car 6 from input track to S2
- ix. Car 9 from input track to S1
- x. Car 5 from input track to back of output track
- xi. Car 6 from S2 to back of output track
- xii. Car 7 from S2 to back of output track
- xiii. Car 8 from S2 to back of output track
- xiv. Car 9 from S1 to back of output track

b.

123498765

Car 5 from input track to S1

Car 6 from input track to S2

Car 7 from input track to S3

Error: If we move Car 8 to the back of the output track, the rearranging fails because 8 will be first instead of 1. If we move Car 8 to one of the holding tracks, 8 must come before either 5, 6, or 7 in the output sequence, which is a violation of the sequential order we are trying to produce. **Consequently, this train of length 9 cannot be rearranged in increasing order (987654321) using 3 holding tracks.**