

Carson Kraft  
Cck2127  
Data Structures  
Paul Blaer

1.

```
public static void printLots(Collection<Integer> l, Collection<Integer> p){
```

```
    Iterator<Integer> itP = p.iterator();  
    Iterator<Integer> itL = l.iterator();
```

```
    int x = itL.next();  
    int lastPoint = 0;
```

```
    while(itP.hasNext()){
```

```
        int point = itP.next();
```

```
        for (int i=0; i < point - lastPoint; i++){
```

```
            x = itL.next();
```

```
        }
```

```
        System.out.println(x);
```

```
        lastPoint = point;
```

```
    }
```

```
}
```

1.

```
Iterator<Integer> itL1 = L1.iterator();  
Iterator<Integer> itL2 = L2.iterator();
```

```
Int x = itL1.next();  
Int y = itL2.next();
```

```
While L1 has next(
```

```
    If(x = y)
```

```
        Add x to new list;
```

```
        X =itL1.next();
```

```
        If(itL2.hasNext()){
```

```
            Y = itL2.next();
```

```

        }
        Else{ break;}
    }
    if(x > y){

        if(itL2.hasNext()){
            Y = itL2.next();
        }
        Else{ break;}

    }

    else {
        X =itL1.next();
    }
}

```

2. TopA = 0;  
TopB = a.length - 1;

```

pushA(int x) {

    if(topA = topB + 1) {
        the array is full → overflow error;
    }

    a[topA] = x;
    top A ++;
}

```

```

popA() {

    int temp = a[topA];
    a[topA] = null;
    topA --;
    return temp;
}

```

```

peakA() {

    int p = a[topA];
    return p;

}

```

```

isEmptyA() {

    if(topA == 0){ return true }
    else{ return false; }
}
sizeA() {

    size = topA -1;
    return size;
}

pushB(int x) {

    if(topA = topB + 1) {
        the array is full → overflow error;
    }

    a[topB] = x;
    topB ++;
}

popB() {

    int temp = a[topB];
    a[topB] = null;
    topB --;
    return temp;
}

peakB() {

    int p = a[topB];
    return p;
}

isEmptyB() {

    if(topB == array.length -1){ return true }
    else{ return false; }
}
sizeB() {

    size = array.length – topB - 1;
    return size;
}

```

3. A)
- 4 to S1
  - 3 to S1
  - 1 to output track
  - 8 to S2
  - 2 to output track
  - 3 to output track
  - 4 to output track
  - 7 to S2
  - 6 to S2
  - 9 to S1
  - 5 to output track
  - 6 to output track
  - 7 to output track
  - 8 to output track
  - 9 to output track
- B) [1, 2, 9, 6, 7, 8, 4, 3]