Jinoh Lee
COMSW3134
Paul Blaer
7 October 2016

Written

Problem 1

```
public void printLots(Collection<AnyType> L, Collection<Integer> P) {
	Integer[] pValues = new Integer[P.size()];
	int counter = 0;
	int counter2 = 0;
	int pValueCounter = 0;

	while (P.iterator().hasNext()) {
		Integer element = P.iterator().next();
		pValues[counter] = element;
		counter++;
	}

	while (L.iterator().hasNext()) {
		Object element2 = L.iterator().next();
		if (counter2 == pValues[pValueCounter]) {
			System.out.println(element2);
			pValueCounter++;
		}
		counter2++;
	}
}
```

Problem 2 (Weiss 3.4)

```
public void method that accepts two list inputs (L1 and L2) {
	create ListIterator itr1 to iterate through L1
	while itr1 hasNext {
		object element = itr1.next();
		if L2.contains(element) {
		System.out.println(element);
		}
	}
}
```

Problem 3 (Weiss 3.24)

```
public class twoStacksOneArray {
	Initialize Integers bottom1, bottom2, top1, top2;
	constructor method for AnyType array (accepts input) {
		array = input;
```

```
        int length = array.length;
        if length%2 == 1 {
                bottom1 = (array.length/2) – 1;
                bottom2 = (array.length/2) + 1;
        } else {
                bottom1 = array.length/2;
                bottom2 = (array.length/2) + 1;
        }
        top1 = bottom1;
        top2 = bottom2;
}

public void push1(AnyType a) {
        if (top1 < 0) {
                throw error to catch out of index
                System.out.println(“Overflow”);
        } else {
                array[top1] = a;
                top1--;
        }
}

public void push2(AnyType a) {
        if(top2 >= length) {
                throw error to catch out of index
                System.out.println(“Overflow”);
        } else {
                array[top2] = a;
                top2++;
        }
}

public AnyType pop1() {
        if (top1 > bottom1) {
                System.out.println(“You can’t pop more in this stack”);
        } else if (top1 == bottom1) {
                temp = array[top1];
                array[top1] = null
                return temp;
        } else {
                temp = array[top1];
                array[top1] = null
                top1++;
                return temp;

}
public AnyType pop2() {
        if (top2 < bottom2) {
```

```
                System.out.println("You can't pop more in this stack");
        } else if (top2 == bottom2) {
                temp = array[top2];
                array[top2] = null;
                return temp;
        } else {
                temp = array[top2];
                array[top2] = null;
                top2--;
                return temp;
        }
    }
}
```

Problem 4

A.

Note: Output is ordered back to front in work to mirror example in problem

1. Push Car 4 to S3 (S3: 4)
2. Push Car 3 to S3 (S3: 3, 4)
3. Enqueue Car 1 to Output (Output: 1)
4. Push Car 8 to S2 (S2: 8)
5. Enqueue Car 2 to Output (Output: 2, 1)
6. Pop Car 3, Enqueue to Output (Output: 3, 2, 1) (S3: 4)
7. Pop Car 4, Enqueue to Output (Output: 4, 3, 2, 1)
8. Push Car 7 to S2 (S2: 7, 8)
9. Push Car 6 to S2 (S2: 6, 7, 8)
10. Push Car 9 to S3 (S3: 9)
11. Enqueue Car 5 to Output (Output: 5, 4, 3, 2, 1)
12. Pop Car 6, Enqueue to Output (Output: 6, 5, 4, 3, 2, 1) (S3: 7, 8)
13. Pop Car 7, Enqueue to Output (Output: 7, 6, 5, 4, 3, 2, 1) (S3: 8)
14. Pop Car 8, Enqueue to Output (Output: 8, 7, 6, 5, 4, 3, 2, 1)
15. Pop Car 9, Enqueue to Output (Output: 9, 8, 7, 6, 5, 4, 3, 2, 1) This is back to front
16. Front to Back Output: 1, 2, 3, 4, 5, 6, 7, 8, 9

B.

9, 1, 8, 2, 7, 3, 6, 4, 5