Sunand Iyer
sri211

HW 2

1)

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

        Iterator itrP = P.iterator();
        while (itrP.hasNext()){
                itrL = L.iterator();
                int index = itrP.next();
                int counter = 0;
                while (itrL.hasNext()){
                        if (counter == index){
                                int output = itrL.next();
                                System.out.println(output);
                        }
                        counter += 1;
                }
        }
}
```

2)

```
public List intersection(List1, List2){

        List output;
        int size1 = List1.size();
        int size2 = List2.size();
        for (int i = 0; i < size1; i++){
                element1 = List1.get(i);
                for (int j = 0; j < size2; j++){
                        element2 = List2.get(j);
                        if (element2 == element1){
                                output.add(element2);
                        }
                }
        }
        //now get rid of duplicates in output
        for (int i = 0; i < output.size(); i++){
                element = output.get(i)
                for (int j = i+1; j < output.size(); j++){
                        if (element == output.get(j)){
                                output.remove(j);
                        }
                }
        }
```

```
        return output;
}
```

The runtime is O(N^2).

3)

```
import java.util.Arrays;

public class TwoStackArray<AnyType>{

        public AnyType[] stackArray;
        public int s1End;
        public int s2End;


        public TwoStackArray(){

                stackArray = (AnyType[]) new Object[100];
                s1End = 0;
                s2End = stackArray.length - 1;

        }

        void pushS1(AnyType y){

                if (s1End == stackArray.length || s1End == s2End + 1){
                        throw StackOverflowError;
                } else {
                        for (int i = s1End-1; i >= 0; i--){
                                stackArray[i+1] = stackArray[i];

                        }
                        stackArray[0] = y;
                        s1End += 1;
                }
        }

        void pushS2(AnyType y){

                if (s2End == 0 || s2End == s1End - 1){
                        throw StackOverflowError;
                } else {
                        for (int i = s2End+1; i <= stackArray.length-1; i++){
                                stackArray[i-1] = stackArray[i];
                        }
                        stackArray[stackArray.length-1] = y;
                        s2End = s2End - 1;
                }
```

```
}

AnyType popS1(){

	if (s1End == 0){
		System.out.println("Empty Stack");
		return stackArray[0];
	} else {
		AnyType output = stackArray[0];
		for (int i = 0; i < s1End-1; i++){
			stackArray[i] = stackArray[i+1];
		}
		stackArray[s1End-1] = null;
		s1End = s1End - 1;
		return output;
	}
}

AnyType popS2(){

	if (s2End == stackArray.size - 1){
		System.out.println("Empty Stack");
		return stackArray[stackArray.size - 1];
	} else {
		AnyType output = stackArray[stackArray.size - 1];
		for (int i = stackArray.length - 1; i > s2End+1; i--){
			stackArray[i] = stackArray[i-1];
		}
		stackArray[s2End+1] = null;
		s2End = s2End + 1;
		return output;
	}

}

AnyType peekS1(){
	if (stackArray[0] == Null){
		System.out.println("Empty Stack");
		return stackArray[0];
	} else {
		return stackArray[0];
	}
}

AnyType peekS2(){
	if (stackArray[stackArray.size - 1] == Null){
		System.out.println("Empty Stack");
		return stackArray[stackArray.size - 1];
	} else {
```

```
                return stackArray[stackArray.size - 1];
            }
        }

        boolean isEmptyS1(){
            return stackArray[0] == Null;
        }

        boolean isEmptyS2(){
            return stackArray[stackArray.size - 1] == Null;
        }

        int sizeS1(){
            if (stackArray[0] == Null){
                return 0;
            } else {
                return s1End;
            }
        }

        int sizeS2(){
            if (stackArray[stackArray.size - 1] == Null){
                return 0;
            } else {
                return stackArray.size - s2End -1;
            }
        }



}
```

4)

a. Steps:

1) Push car 4 onto S1
2) Push car 3 onto S1
3) Push car 1 to output track
4) Push car 8 onto S2
5) Push car 2 to output track
6) Pop car 3 from S1 and push it to output track
7) Pop car 4 from S1 and push it to output track
8) Push car 7 onto S2
9) Push car 6 onto S2
10) Push car 9 onto S3
11) Push car 5 onto moving track
12) Pop car 6 from S2 and push it to output track
13) Pop car 7 from S2 and push it to output track

14) Pop car 8 from S2 and push it to output track
15) Pop car 9 from S3 and push it to output track

b. 139827654