Yilun Ying
Yy2658
Homework 2

Problem 1:

```
private static <AnyType> void printLots(List<AnyType> L, List<Integer> P) {
		java.util.Iterator<AnyType> n = L.iterator();
		java.util.Iterator<Integer> o = P.iterator();

		int previousPosition = -1; //Notes the previous position
		int numToAdvance; //Notes how many positions to advance

		while (o.hasNext()) {//Iterates through the list P
			Integer position = o.next();
			// Reads the position written in list P
			AnyType content = null;
			//Stores the content to be retrieved from L
			numToAdvance = position - previousPosition;
			//Calculates how many positions to iterate
			previousPosition = position;//Sets the previous position

				for (int i = 0; i < numToAdvance; i++) {//advances L
					if (n.hasNext()) {// in case of index out of bounds
					content = n.next();
					} else {
						System.out.println("No such element");
						return;
					}
				}
				System.out.println(content);
		}
	}
```

Problem 2

```
public static <AnyType extends Comparable<AnyType>> void
checkIntersect(List<AnyType> L1, List<AnyType> L2) {
        for (int i = 0; i < L1.size(); i++) {
                AnyType content;
                content = L1 at position i;
                for (int j = 0; j < L2.size(); j++) {
                        if (L2 at position j is the same as content) {
                                Print content
                        }
                }

        }

}
```

Problem 3

```
public class TwoStackOneArray<AnyType> {

	AnyType[] array;
	private int stack1Empty;
	private int stack2Empty;

	public TwoStackOneArray() {//initializes the class
		array = (AnyType[]) new Object[However long it should be];
		stack1Empty = 0;//stack 1 starts from front
		stack2Empty = array.length-1;//stack 2 starts from back
	}

	public void stack1Push(AnyType e){//pushes element into stack 1
		if(stack1Empty==stack2Empty+1){
			System.out.println("Error - Array full");
		} else {
		array[stack1Empty] = e;
		stack1Empty++;
		}
	}

	public void stack2Push(AnyType e){//pushes element into stack 2
		if(stack1Empty==stack2Empty+1){
			System.out.println("Error - Array full");
		} else {
		array[stack2Empty] = e;
		stack2Empty--;
		}
	}

	public AnyType stack1Pop(){//pops element from stack 1
		if(stack1Empty==0){
			return null;
		} else {
			stack1Empty--;
			return array[stack1Empty];
		}
	}

	public AnyType stack2Pop(){//pops element from stack 2
		if(stack2Empty==array.length-1){
			return null;
```

```
            } else {
                stack2Empty++;
                return array[stack2Empty];
            }
        }

        public AnyType stack1Peek(){//peeks what's at the top of stack 1
            if(stack1Empty==0){
                return null;
            } else {
                return array[stack1Empty-1];
            }
        }

        public AnyType stack2Peek(){//peeks what's at the top of stack 2
            if(stack2Empty==array.length-1){
                return null;
            } else {
                return array[stack2Empty+1];
            }
        }



}
```

Problem 4

a)
move 4 from input to S1
move 3 from input to s1
move 1 from input to output track
move 8 from input to s2
move 2 from input to s1
move 2 from s1 to output track
move 3 from s1 to output track
move 4 from s1 to output track
move 7 from input to s2
move 6 from input to s2
move 9 from input to s1
move 5 from input to output track
move 6 from s2 to output track
move 7 from s2 to output track
move 8 from s2 to output track
move 9 from s1 to output track

b) [1,3,5,7,9,8,6,4,2]