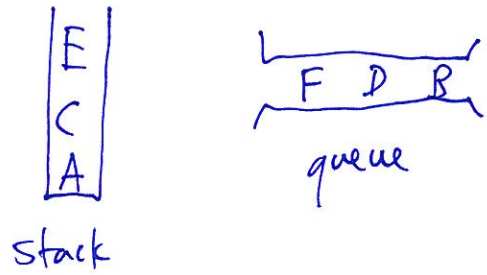
Question 4. [10 points] Consider the following method:

```
public static void mystery(List<String> list) {
    Stack<String> stack = new Stack<String>();
    Queue<String> queue = new LinkedList<String>();

    Iterator<String> i = list.iterator();
    while (i.hasNext()) {
        stack.push(i.next());
        if (i.hasNext()) {
            queue.add(i.next());
        }
    }

    while (!stack.isEmpty()) {
        System.out.println(stack.pop());
    }

    while (!queue.isEmpty()) {
        System.out.println(queue.remove());
    }
}
```



What output is printed by the following code?

```
List<String> coll = Arrays.asList("A", "B", "C", "D", "E", "F");
Q4.mystery(coll);
```

E
C
A
} from stack

B
D
F
} from queue

Question 5. [10 points] Complete the `containsDuplicates` method below. It takes a `Collection` of elements of type `E` as a parameter, and returns true if the collection contains any duplicate elements, or false if the collection does not contain any duplicate elements. You can assume that the type `E` implements `Comparable<E>`.

Requirement: The method should complete in $O(N \log N)$ running time (or $O(N)$ running time), where N is the number of elements in the collection.

Here are some JUnit tests showing the expected behavior of the method:

```
List<String> listA = Arrays.asList("A", "B", "C", "A", "D");
List<String> listB = Arrays.asList("P", "V", "Z", "Y");

assertTrue(Q5.containsDuplicates(listA));
assertFalse(Q5.containsDuplicates(listB));
```

Hint: What kind of collection is useful for detecting duplicate values?

```
public static<E extends Comparable<E>>
boolean containsDuplicates(Collection<E> coll) {
    TreeSet<E> s = new TreeSet<E>();
    s.addAll(coll);
    return s.size() < coll.size();
}
```