CSE 122 Section Homework - Section 16

Hello everyone! Today's section homework will begin our final exam review, covering `ArrayLists`, Stacks, and Queues.

1. Consider the method below.

```
public static List<Integer> mystery(int[][] data) {
    List<Integer> result = new ArrayList<>();
    for (int i = 0; i < data.length; i += 2) {
        for (int j = data[i].length - 1; j >= 0; j--) {
            data[i][j]++;
            result.add(data[i][j]);
        }
    }
    return result;
}
```

For each 2D array below, indicate in the right hand column what values would be stored in the list returned by the method `mystery` if the array in the left-hand column is passed as the parameter `data`. List elements should be listed as a comma-separated, bracketed list (ex: [1, 2, 3, 4]).

Input 2D Array	Contents of List Returned

`[[1, 2, 3],`
`[1, 2, 3],`
`[1, 2, 3]]`	

`[[3, 6],` `[4, 11]]`	

`[[2, 7],`
`[6, 1],`
`[91, 8],`
`[77, 7]]`	

Now, describe what this method does.

2. The method `oddsAndEvens` takes in a queue of integers called `numbers` as a parameter and rearranges the order of the values so that all of the odd values appear before the even values. If the number is greater than 20, we do not want it considered in the output. Otherwise, the order of the original queue is preserved. For example, supposed `nums` stores these values:

```
front  [85, 13, 96, 4, 17, 20, 1, 3, 8]  back
```

Then the call to `oddsAndEvens(numbers);` should rearrange the queue to store the following sequence of integers:

```
front  [13, 17, 1, 3, 4, 20, 8]  back
```

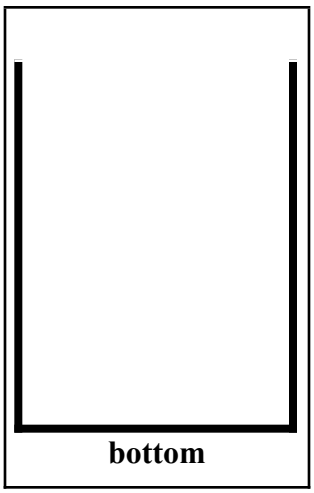
Notice that all of the odd integers appear at the front of the queue followed by the evens. Also, no numbers larger than 20 are included and the order of odds and evens are the same as the original queue.

```
1   public static void oddsAndEvens(Queue<Integer> numbers) {
2       Stack<Integer> storage = new Stack<>();
3       int size = numbers.size();
4       for (int i = 0; i < size; i++) {
5           int currNum = numbers.remove();
6           if (currNum <= 20) {
7               if (currNum % 2 == 1) {
8                   numbers.add(currNum);
9               } else {
10                  storage.push(currNum);
11              }
12          }
13      }
14      // A
15      int storedSize = storage.size();
16      for (int i = 0; i < storedSize; i++) {
17          int storedNum = storage.pop();
18          numbers.add(storedNum);
19      }
20      // B
21      int newSize = numbers.size();
22      for (int i = 0; i < newSize; i++) {
23          int currNum = numbers.remove();
24          if (currNum % 2 == 1) {
25              numbers.add(currNum);
```

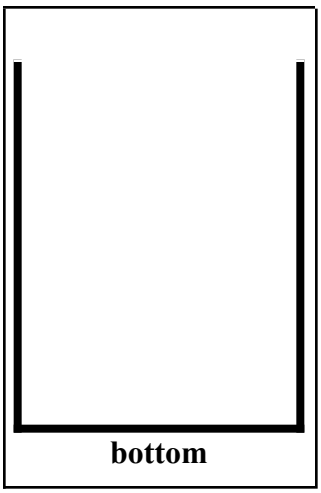
```
26          } else {
27              storage.push(currNum);
28          }
29      }
30      // C
31      int newStoredSize = storage.size();
32      for (int i = 0; i < storedSize; i++) {
33          int storedNum = storage.pop();
34          numbers.add(storedNum);
35      }
36  }
37
```

Draw the contents of the stack `storage` when the code reaches points A, B, and C.

i. A



ii. B



iii. C