CSE143 Section #1 Problems

We ended the first lecture with the following version of ArrayIntList:

```
    public class ArrayIntList {
        private int[] elementData; // list of integers
        private int size;          // current number of elements in the list

        // constructs a list with a capacity of 100
        public ArrayIntList() {
            elementData = new int[100];
            size = 0;
        }

        // returns a comma-separated, bracketed version of the list
        public String toString() {
            if (size == 0) {
                return "[]";
            } else {
                String result = "[" + elementData[0];
                for (int i = 1; i < size; i++) {
                    result += ", " + elementData[i];
                }
                result += "]";
                return result;
            }
        }

        // appends the given value to the end of the list
        public void add(int value) {
            elementData[size] = value;
            size++;
        }
    }
```

1. Array simulation. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
        public static void mystery(int[] list) {
            for (int i = 1; i < list.length; i++) {
                list[i] = list[i] + list[i - 1];
            }
        }
```

   In the left-hand column below are specific lists of integers. You are to indicate in the right-hand column what values would be stored in the list after method mystery executes if the integer list in the left-hand column is passed as a parameter to mystery.

Original List	Final List
{8}	____________________________
{6, 3}	____________________________
{2, 4, 6}	____________________________
{1, 2, 3, 4}	____________________________
{7, 3, 2, 0, 5}	____________________________

2. Write a new method for the ArrayIntList class called indexOf that returns the index of a particular value in the list. The method should return the index of the first occurrence of the target value in the list. If the value is not in the list, it should return -1. For example, if a variable called list stores the following values:

   ```
   [42, 7, -9, 14, 8, 39, 42, 8, 19, 0]
   ```

   then the call list.indexOf(8) should return 4 because the index of the first occurrence of the value 8 in the list is at index 4. Notice that we are using 0-based indexing. The call list.indexOf(2) should return -1 because the value 2 is not in the list.

3. Write a new method for the ArrayIntList class called stutter that doubles the size of the list by replacing every integer in the list with two of that integer. For example, if a variable called list stores the following:

   ```
   [1, 8, 19, 4, 17]
   ```

   and we make the following call:

   ```
   list.stutter();
   ```

   then it should store the following sequence of integers after the call:

   ```
   [1, 1, 8, 8, 19, 19, 4, 4, 17, 17]
   ```

4. Write a new method for the ArrayIntList class called remove that takes an integer index and that removes the value at the given index, shifting subsequent values to the left. For example, if a variable called list stores the following values:

   ```
   [3, 19, 42, 7, -3, 4]
   ```

   and we make the following call:

   ```
   list.remove(1);
   ```

   then it should store the following sequence of integers after the call:

   ```
   [3, 42, 7, -3, 4]
   ```

5. Write a new method for the ArrayIntList class called add that takes an integer index and a value to add and that inserts the given value at the given index, shifting subsequent values to the right. For example, if a variable called list stores the following values:

   ```
   [3, 19, 42, 7, -3, 4]
   ```

   and we make the following call:

   ```
   list.add(2, 17);
   ```

   then it should store the following sequence of integers after the call:

   ```
   [3, 19, 17, 42, 7, -3, 4]
   ```