CSSE 230 In-class exercise Day 1 Names _____ _____ Section _____

GrowableArrays activity.
Work with a partner. If an odd number of students are present, one group may have three people.

This activity deals with the overhead of adding many elements to a "growable array" data structure, under different growing strategies:

(A) "Add-one" strategy: [new capacity] = [old capacity] + 1
(B) "Doubling" strategy: [new capacity] = 2 * [old capacity]

Code for these strategies is given on the second page.

Let N be the total number (called `NUM_TO_ADD` in the code) of items that are added to a growable array that is initially empty with a capacity of 4. For simplicity**, assume N is one more than a power of 2**: say, $N-1 = 2^k$ for some number k.

1. In the table, tally the number of **writes to the array** that happen when the i^{th} element is added to the array, where *i* ranges from 0 to N–1. A few are completed for you.

i	(A) #writes	(B) #writes
0	1	1
1	1	1
2	1	1
3	1	1
4	1+4	1+4
5	1+5	1
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
…		
N–2		
N–1		

2. Over the N adds, what is the total number of writes for strategy (A)? Your answer should be in closed form (no sum notation).

3. Over the N adds, what is the total number of writes for strategy (B)? Your answer should be a closed-form function of N only.

4. Consider your answers from problems 2 and 3 as the "total cost of adding N items" under each strategy. What then is the *amortized* cost of adding a single item… [Hint: divide your previous answers by N]

   for the add-one strategy (A)? for the doubling strategy (B)?

5. For which strategy does the Big-O *amortized cost* differ from the Big-O *worst-case cost* of an add? Explain.

6. Is there ever a situation where the add-one strategy might be preferable to the doubling strategy? Explain.

```
public class GrowableArray {
	int[] array;
	int size;
	int capacity;
	static final int INITIAL_CAPACITY = 5;

	GrowableArray() {
		this.capacity = INITIAL_CAPACITY;
		this.array = new int[this.capacity];
		this.size = 0;
	}

	public void addToEnd(int item) {
		if (size == capacity) {
                                        // Use one of the following lines:
			resize(capacity + 1); // (A) resize by add-one strategy
                                        //                OR
//			resize(2 * capacity); // (B) resize by doubling strategy
		}
		this.array[size] = item;
		size++;
	}

	private void resize(int newCap) {
		int[] newArray = new int[newCap];
		for (int i = 0; i < this.size; i++) {
			newArray[i] = this.array[i];
		}
		this.array = newArray;
		this.capacity = newCap;
	}

	// Main method for testing basic functionality.
	public static void main(String[] args) {
		int NUM_TO_ADD = 10000;
		GrowableArray ga = new GrowableArray();
		for (int i = 0; i < NUM_TO_ADD; i++) {
			ga.addToEnd(i);
		}
	}
}
```