Quiz 18: Arrays SOLUTION

CSCI 110 Section 1

Wednesday, October 12, 2016

1) Write a function that returns the sum of all of the elements of an integer array. [30 points]

Return type? **int**, since we want to return one value: the sum
Parameters? the array of int we want to sum

```
int sumArray(int[] arr) {
      int sum = 0;
      for (int i = 0; i < arr.length; i++) {
            sum += arr[i];
      }
      return sum;
}
```

2) Write a function that fills an integer array with the numbers 1 through 10, then returns it. [30 points]

Return type? **array of int**
Parameters? nothing, we're always going to return an array with the numbers 1 through 10

```
int[] getOneThroughTenArray() {
      int[] result = new int[10];
      for (int i = 0; i < 10; i++) {
          result[i] = i+1;
      }
      return result;
}
```

Here's another solution, which I prefer because it's immediately obvious what this function does.

```
int[] getOneThroughTenArray() {
      int[] result = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
      return result;
}
```

3) Write a function that fills an integer array starting at 1 and ending at a user-specified upper bound (inclusive) [40 points]

Return type? **array of int**
Parameters? an int, the upper bound

```
int[] makeArray(int bound) {
	int[] result = new int[bound];
	for (int i = 0; i < bound; i++) {
		result[i] = i+1;
	}
	return result;
}
```

4) Write a function that fills an integer array starting at a user-specified lower bound and ending at a user-specified upper bound (inclusive if it happens to end on the upper bound), skipping every other number. [extra credit, 25 points]

Return type? **array of int**
Parameters? ints for the lower and upper bounds

For this one, we use two variables as we're going through:

- i, which starts at 0 and increments by 1. This is the location in the array where we're putting a number.
- curr, which starts at the user-provided lower bound and increments by 2. This is the current value we're putting into the array at index i.

The most important part of the solution below is figuring out how big the array should be. It's subtle, but we use integer division to automatically round down when the difference between upper and lower is an odd number.

```
int[] makeEveryOtherArray(int lower, int upper) {
	int[] result = new int[(upper - lower) / 2 + 1];
	int curr = lower;
	for (int i = 0; i < result.length; i++) {
		result[i] = curr;
		curr += 2;
	}
	return result;
}
```