CS 61B Discussion 3: [A, r, r, a, y, s] Fall 2014

1 Let's start with some syntax.

What does the following code print?

```
int[] x = {1, 2, 3, 4, 5};
int[] y = x;
y[2] = 7;
int[] z = new int[3];
z[2] = y[3];
int[][] a = new int[3][];
a[0] = x;
a[1] = y;
System.out.println("A: " + Arrays.deepToString(a));
System.out.println("Z: " + Arrays.toString(z));
```

```
[[1, 2, 7, 4, 5], [1, 2, 7, 4, 5], null]
[0, 0, 4]
```

2 Debugging is good for your health

The following code is broken. Please identify and fix the errors.

a)

```
int[] a;
a = {1, 2, 3};
int[] z = {4, 5, 6};
int[] y;
y = new int[]{7, 8, 9};
```

You can't initialize an array with curly braces after you've declared the variable. An error would occur when you try to declare a = {1, 2, 3}.

```
int[] count = {0, 2, 3, 5};
for (; count[0] < count[3]; count[0] = count[0] + 1) {
    System.out.println(count[count[0]]);
}
```

You get an out of bounds error because the for loop condition is checking against count[3], which is larger than the array length. You must use count.length for the correct functionality.

3 Filling in the blanks

Fill in the blanks to complete the following methods.

a)

```
/** Given an array A of at least 1 element, return the
  * average of all the elements.
  */
public static double average(double[] A) {
    double sum = 0.0;

    for (int i = 0; i < A.length; i += 1) {
        sum += A[i];
    }
    return sum/A.length;
}
```

b)

```
import static java.lang.Math.max;
import static java.lang.Math.min;
/** Given an array A, return a 2 element array B where
  * B[0] is the minimum element of A and B[1] is the
  * maximum element of A.
  */
public static int[] minMax(int[] A) {
    int maxVal = Integer.MIN_VALUE;
    int minVal = Integer.MAX_VALUE;
    int[] B = new int[2];

    for (int i = 0; i < A.length; i+= 1) {
        maxVal = max(maxVal, A[i]);
        minVal = min(minVal, A[i]);
    }
    B[0] = minVal;
    B[1] = maxVal;
    return B;
}
```

4 GoogitterBook Engineering Interview

Welcome to GoogitterBook, I hear you're interested in a position here. First, let's see if you can program. Given an integer x and a SORTED array A[] of N distinct integers, design an algorithm to find if there exists indices i and j such that A[i] + A[j] == x.

a) Let's start with the naive solution.

```
public static boolean findSum(int[] A, int x) {
    for (int i = 0; i < A.length; i++) {
        for (int j = 0; j < A.length; j++) {
                if (A[i] + A[j] == x) {
                    return true;
                }
        }
    }
    return false;
}
```

b) Can we do this faster? Hint: Does order matter here?

```
public static boolean findSumFaster(int[] A, int x) {
    int left = 0;
    int right = A.length - 1;
    while (left < right) {
        if (A[left] + A[right] == x) {
            return true;
        } else if (A[left] + A[right] < x) {
            left++;
        } else {
            right--;
        }
    }
    return false;
}
```

c) Bonus for Bosses Very good, now let's add another dimension to this. Given an array A[] of N distinct integers, there exist indices i, j, and k such that A[i] + A[j] + A[k] == 0. Design an algorithm to solve this problem. Hint: Use your answer to part b.

```
public static boolean threeSum(int[] A) {
    int j, k;
    for (int i = 0; i < A.length; i++) {
        int j = i + 1;
        int k = A.length - 1;
        while (j < k) {
            if (A[i] + A[j] + A[k] == 0) {
                return true;
            } else if (A[i] + A[j] + A[k] > 0) {
                k--;
            } else {
                j++;
            }
        }
    }
    return false;
}
```