Introduction to Computer Science I
Fall 2017
Mid-term exam — Solutions

1. Questions: Provide short answers (one to three sentences) to each of the following questions:
   (a) If a Java program compiles with no errors, is it then guaranteed to run correctly? Justify your answer.
   (b) Consider the following declaration: `int[] x;`
   What, specifically, is in the space named `x`?
   (c) What, in Java, is an *expression*?

   Answers:

   (a) **No.** Compilers detect errors in *syntax* and in *type usage*, and prevent the possible use of uninitialized spaces. At execution time (when the program runs), *semantic* errors that no compiler can detect may appear. An program that compiles and runs, but which produces incorrect output, serves as an example of this difference.
   (b) The same named `x` contains *a pointer to an array of integers* (`int`).
   (c) An expression, in Java, is any code that, when executed, yields some resultant value. Examples of expressions are: *constants*, which yield their face value; *variables*, which yield the value stored in the space tied to that name; *arithmetic operations*, which produce the result of that arithmetic calculation; and *method calls* that return some type of data.

2. QUESTION: Complete this *truth table* of Java's Boolean logic operators:

`a`	`b`	`a && b`	`a || b`	`a == b`	`a != b`
F	F	F	F		
F	T	F	T		
T	F	F	T		
T	T	T	T		

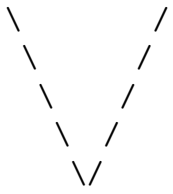
ANSWER:

`a`	`b`	`a && b`	`a || b`	`a == b`	`a != b`
F	F	F	F	T	F
F	T	F	T	F	T
T	F	F	T	F	T
T	T	T	T	T	F

Thus, the *inequality* operator (`!=`) is also the *exclusive or* operator![1]

[1]Not part of the question, but an interesting observation.

3. QUESTION: Write a method named `printBigV` that, when passed a size (in this example, `5`), prints the following pattern:



ANSWER:

```
public static void printBigV (int size) {
    int row = 0;
    while (row < size) {
        int leadSpaces   = row;
        int middleSpaces = (size - row - 1) * 2;
        while (leadSpaces > 0) {
            System.out.print(' ');
            leadSpaces = leadSpaces - 1;
        }
        System.out.print('\\');
        while (middleSpaces > 0) {
            System.out.print(' ');
            middleSpaces = middleSpaces - 1;
        }
        System.out.println('/');
        row = row + 1;
    }
}
```

4. QUESTION: Complete the following method such that it changes the given array of `char` by changing all lowercase letters into uppercase ones. [Hint: Recall that each character is really a number, internally, and that the characters `A` to `Z` are represented with 26 values in a row, as are characters `a` to `z` by a different 26 contiguous numbers. You do **not** need to know what those specific numeric values are.]

```
public static void toUpperCase (char[] msg)
```

ANSWER:

```
public static void toUpperCase (char[] msg) {
    int i = 0;
    while (i < msg.length) {
        if ('a' <= msg[i] && msg[i] <= 'z') {
            msg[i] = (char)(msg[i] - ('a' - 'A'));
        }
        i = i + 1;
    }
}
```

5. QUESTIION: What is the output of this program, `Conditionals`, when it is run?

```
public class Conditionals {

    public static void main (String[] args) {
        foo(-5);
        bar(-5);
    }

    public static void foo (int x) {
        if (x < 0) {
            System.out.println("Message 1: " + x);
            x = -x;
        }
        if (x >= 0) {
            System.out.println("Message 2: " + x);
            x = -x;
        }
        System.out.println("Message 3: " + x);
    }

    public static void bar (int x) {
        if (x < 0) {
            System.out.println("Message A: " + x);
            x = -x;
        } else {
            System.out.println("Message B: " + x);
            x = -x;
        }
        System.out.println("Message C: " + x);
    }

}
```

ANSWER:

```
Message 1: -5
Message 2: 5
Message 3: -5
Message A: -5
Message C: 5
```