CIS 110 Fall 2014 — Introduction to Computer Programming
8 Oct 2014 — Midterm Exam
Answer Key

0.) THE EASY ONE (1 point total)

Check cover sheet for name, recitation #, PennKey, and signature.

1.) TYPES AND VALUES (9 points total)

Fill in the data type and final value of the variable `a`. (Assume `a` has been declared with the appropriate data type.) Write "**CE**" as the data type if the statements will cause compiler error, or "**RE**" if they will cause a run-time error. Give the reason for the error in the third column. The first row has been filled in for you.

	Type of `a`	Value of `a`/Error explanation
`_____ a = 2;`	`int`	2

`_____ a = false;`
`a = a || (true && !false);`
	`boolean`	`true`

`double x = 3;`
`x--;`
`_____ a = x;`
	double	`2.0` -or- `2`

`String x = "oneten";`
`_____ a = x.charAt('4');`
	RE-- `or` --CE	`'4' == 42`: out of bounds
		`-- or --`
		argument should be `int` not `char`

`double x = 4;`
`_____ a = x.pow(.5);`
	compiler error	Use `a = Math.pow(x, .5);`

`int[] x = { 3, 1, 4, 1 };`
`_____ a = x[x[0]];`
	`int`	1

`String x = "oneten";`
`_____ a = String.length(x);`
	compiler error	Use `a = x.length();`

`_____ a = "1";`
`a = a + a;`
`a += a;`
	String	`"1111"` -or- `1111`

`_____ a = 3;`
`a %= 2;`
	int	`1`

`int x = 15;`
`int y = 2 * x / 2;`
`_____ a = x == y;`
	`boolean`	`true`

2.) HORTON HIRES A WHO (12 points total)
The Horton School of Funny Business has hired you, a recent CIS 110 graduate, to rewrite its grade calculation software. Your first task is to write a function **`computeLetterGrades()`** that takes an array of percentage scores (between 0 and 100, rounded to the nearest 10th of a percent), and returns an array of letter grades corresponding to the table below. The function does not need to do any error checking. You may use any of the functions from Java's `Math` library.

Percentage	Grade
0 – 49.9	B-
50 – 59.9	B
60 – 69.9	B+
70 – 79.9	A-
80 – 100	50% chance of A, 50% chance of A+

Fill in the blanks below to complete the `computeLetterGrades()` function:

```
public static String[] computeLetterGrades(double[] grades) {
    String[] letterGrades = new String[grades.length];

    for (int i = 0; i < grades.length; i++) {
        if (grades[i] < 50) {
            letterGrades[i] = "B-";
        } else if (grades[i] < 60) {
            letterGrades[i] = "B";
        } else if (grades[i] < 70) {
            letterGrades[i] = "B+";
        } else if (grades[i] < 80) {
            letterGrades[i] = "A-";
        } else if (Math.random() < 0.5) {
            letterGrades[i] = "A";
        } else {
            letterGrades[i] = "A+";
        }
    }

    return letterGrades;
}
```

3.) DEBUGGING (10 points total)

Benedict has written a program to fill an array of length `args[0]` with random integers from 0 to 9 (including 0 and 9), then print out the average of all even numbers in the array. But as you know from lecture, he is prone to mistakes. Find the 8 buggy lines in his program, and write the **line numbers and complete, corrected lines** on the next page. (Some lines may have more than one bug, but they still count as a single, buggy line.) To delete a line, write the line number and "**`Delete line.`**" To insert a line, write the line numbers **on either side** of the point where you will insert your code, and write "**`Add line:`**" and your line of code. The first buggy line has been corrected for you, so you have 7 more to find.

```
00: public class EvenAverage
01:     public static void main(String[] args) {
02:         int fly = Integer.parseInt(args[0]); // number of array entries
03:         int[] tic;                           // array for random numbers
04:
05:         // fill in random numbers
06:         for (gnat = 0; gnat < 10; gnat++) {
07:            tic[gnat] = 10 * Math.random();
08:            gnat = gnat + 1;
09:         }
10:
11:         int louse = 0;      // loop index for inspecting values
12:         int spider = 0;     // sum of even values
13:         int centipede = 0; // number of even values
14:
15:         // find sum of even values and number of even values
16:         while (louse < fly) {
17:             if tic[louse] % 2 = 0 {
18:                 spider += tic[louse];
19:                 centipede++;
20:             }
21:         }
22:
23:         System.out.println("Average of even numbers = " +
24:                                    spider / centipede);
25:     }
26: }
```

Bug 0: Line 00: `public class EvenAverage {`
Bug 1: Line 03: `int[] tic = new int[fly];`
Bug 2: Line 06: `for (int gnat = 0; gnat < fly; gnat++) {`
Bug 3: Line 07: `tic[gnat] = (int) (10 * Math.random());`
Bug 4: Line 08: Delete line.
Bug 5: Line 17: `if (tic[louse] % 2 == 0) {`
Bug 6: 20-21: Insert line: `louse++;`
Bug 7: Line 24: `spider / (double) centipede);`

4.) ALEXANDER THE GREAT (16 points total)
Your friendly TAs Alex Brashear, Alex Kornhauser, and Alex Whitaker were recently working on a class project together. Each TA wrote one function, which he named after himself. The program works, but we're having trouble figuring out what it does. Trace through the program and record the values of `a`, `b`, and `c` in the order that each comment of the form **`/* LINE XX */`** is reached. Assume the program is run with the arguments `1 1`. Write your answers on the following page. The first one has been filled in for you.

```
public class ProjectAlexander {
    public static int alex(boolean c, int a, double b) {
        c = c || (a >= b);
        /* LINE 4 */
        a = alex(a * 2, !c, b * 4);
        /* LINE 6 */
        c = a > b;
        /* LINE 8 */
        b = alex(a * 3, b * 5, !c);
        /* LINE 10 */
        c = !c;
        /* LINE 12 */
        return (int) (b / a);
    }

    public static int alex(int a, boolean c, double b) {
        /* LINE 17 */
        return (int) b + a;
    }

    public static int alex(int a, double b, boolean c) {
        /* LINE 22 */
        return (int) b - a;
    }

    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        double b = Double.parseDouble(args[1]);
        boolean c = false;
        /* LINE 30 */
        a = alex(c, a, b);
        /* LINE 32 */
    }
}
```

```
1: Line  4: a = 1, b = 1.0, c = true      5: Line 22: a = 18, b =   5.0, c = false
2: Line 17: a = 2, b = 4.0, c = false     6: Line 10: a =  6, b = -13.0, c = true
3: Line  6: a = 6, b = 1.0, c = true      7: Line 12: a =  6, b = -13.0, c = false
4: Line  8: a = 6, b = 1.0, c = true      8: Line 32: a = -2, b =   1.0, c = false
```

5.) CODING (20 points total)

For the two code writing questions below, write only the functions. Do not write the class statement. You are welcome to write comments, but it is not required. You may use any of the `Math` functions (`Math.sqrt()`, `Math.min()`, `Math.abs()`, etc.) and any of the string functions (`s.equals()`, `s.compareTo()`, etc.).

5.1) (10 points) Write a function **`sameSign()`** that takes an array of integers and returns `true` if they are all greater than zero or all less than zero, and `false` otherwise. If the array is `null` or empty, return `true`.

```
public static boolean sameSign(int[] arr) {
    if (arr == null) return true;
    for (int i = 0; i < arr.length; i++)
        if (arr[0] * arr[i] <= 0) return false;
    return true;
}

-- or --

public static boolean sameSign(int[] arr) {
    if (arr == null)     return true;
    if (arr.length == 0) return true;
    if (arr[0] < 0) {
        for (int i = 1; i < arr.length; i++)
            if (arr[i] >= 0) return false;
    } else {
        for (int i = 0; i < arr.length; i++)
            if (arr[i] <= 0) return false;
    }

    return true;
  }

-- or --

public static boolean sameSign(int[] arr) {
    if (arr == null || arr.length == 0) return true;
    int pos = 0, neg = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] > 0) pos++;
        if (arr[i] < 0) neg++;
    }

    return (pos == a.length || neg == a.length);
}
```

5.2) (10 points) Write a function **`isSorted()`** that uses `sameSign()` to take an array of strings and returns `true` if they are sorted in alphabetical or reverse alphabetical order and every string is different. Use the `compareTo()` function to determine "alphabetical order." Otherwise return `false`.

If the array is `null` or empty, return `true`. Assume that none of the individual strings is `null`. (Recall that `s.compareTo(t)` returns a positive number if `s` comes after `t`, a negative number if it comes before `t`, and 0 if they are the same.)

```
public static boolean isSorted(String[] arr) {
    if (arr == null) return true;
    if (arr.length == 0) return true;

    int[] sorted = new int[arr.length - 1];
    for (int i = 0; i < sorted.length; i++)
        sorted[i] = arr[i].compareTo(arr[i + 1]);

    return sameSign(sorted);
}
```

6.) TWENTY QUESTIONS (5 points total)
Your friend tells you she has thought of a number between 1 and 400. You would like to guess this number in the least number of tries. Each time you guess a number your friend says 'too high', 'too low' or tells you that you have the right number.

6.1) (2 points) If you play this guessing game correctly then, regardless of the number your friend chooses, you can find it in less than N guesses. What is the smallest possible value of N?

(a) 200

(b) 400

(c) 10

(d) $\boxed{\textbf{9}}$

(e) 8

6.2) (2 points) In **thirty words or less**, explain how you will always find your friend's number in this many guesses.
Try 200. Then 100 or 300, and so on until only one number remains.

6.3) (1 point) What is the name of the algorithm you used? binary search