CIS 110 — Introduction to Computer Programming

12 February 2013 — Midterm

Answer Key

Miscellaneous

0. (1 point)

(a) Write your name, recitation number, and PennKey (username) on the front of the exam.

(b) Sign the certification that you comply with the Penn Academic Integrity Code

Types and Values

1. (9 points) Give the type and value for each of the following expressions and variable declarations. When there is more than one statement, give the type and value that the *last* statement computes. If the code would result in a syntax error, write "N/A" in the first column, and give the reason for the error in the second column. You do not need to write the exact error message.

	Type	Value
`6 / 4`	`int`	1
`String s = "Hello"; s.length;`	N/A	The correct syntax is `s.length()`
`0.0 / 0`	`double`	NaN
`"Hello " + 4 + 3`	`String`	`"Hello 43"`
`!!!false && !!true`	`boolean`	`true`
`String s = StdOut.print("Hello");`	N/A	StdOut.print is a `void` function, cannot assign to a `String`
`Math.pow(5, 2)`	`double`	25.0
`6++`	N/A	Cannot increment a literal value
`char[] arr = {'5', '3'}; arr[2];`	N/A	array index out of bounds

Find the Bugs

2. (10 points) The following program is supposed to print out the second-largest double value specified on the command line, or `-Infinity` if less than two numbers are given. But Hanssen didn't test it before sending it to Dr. Brown, and it's full of bugs. Write the line number and **corrected line of code** for ten bugs in the program. Assume that all arguments represent valid `double` values and that all the values are different.

Write your answers on the following page.

```
 1 public static void class SecondBiggest {
 2     public static void main(String[] args) {
 3       int N = args.length;
 4       int[] input = new int[N];
 5       for (int i = 1; i <= N; i++)
 6           input[i] = args[i];
 7       System.out.println(second(input));
 8     }
 9
10     public static double second(double[] arr){
11         double biggest = Double.NEGATIVE_INFINITY;
12         double secondBiggest = Double.POSITIVE_INFINITY;
13         if (arr.length  < 2) return;
14         for (int i = 0, i < arr.length, i++) {
15             double temp = arr[i];
16             if (arr[i] > biggest) biggest = temp;
17         }
18
19         for (int j = 0; j < arr.length; j++) {
20             temp = arr[j];
21             if ((temp > secondBiggest) && (temp != biggest)) {
22                 secondBiggest = temp;
23         }
24         return secondBiggest
25     }
26 }
```

Bugs 1 and 10 were worth 0.5 points, bug 9 was worth 2 points, all others were worth 1 point.

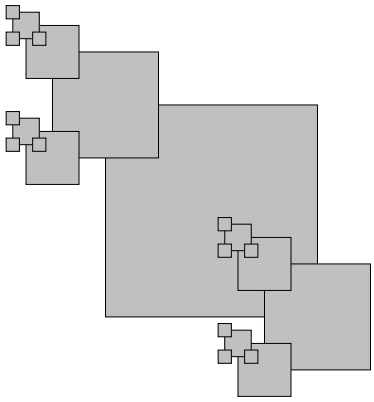
Bug 1: 1: `public class SecondBiggest {`
Bug 2: 4: `double[] input = new double[N];`
Bug 3: 5: `for (int i = 0; i < N; i++)`
Bug 4: 6: `input[i] = Double.parseDouble(args[i])`
Bug 5: 12: `double secondBiggest = Double.NEGATIVE_INFINITY`
Bug 6: 13: `return Double.NEGATIVE_INFINITY`
Bug 7: 14: `for (int i = 0; i < arr.length; i++)`
Bug 8: 20: `double temp = arr[j]`
Bug 9: 21: `if ((temp > secondBiggest) && (temp != biggest))`
Bug 10: 24: `return secondBiggest;`

Recursive Graphics

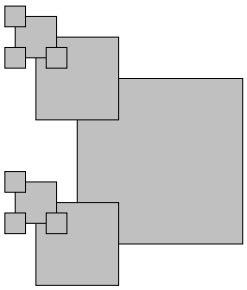
3. (15 points) The figures in this question were all generated using the function below. Assume the `grayStuare()` function draws a gray square with a black outline. For each figure, give values of `r` and `z` that could draw it. Assume the starting values of `x` and `y` are 0.5, and that the starting value of `z` is an integer between 0 and 10.

Hint: Don't worry about scale. We will accept *any* values of `r` and `z` that could generate the figures below at some scale. However, the values we used were all multiples of one eighth.

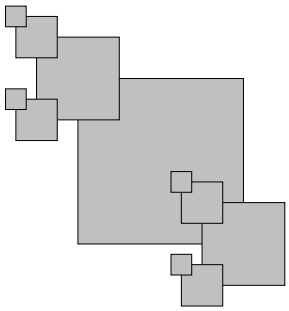
```
public static void draw(double x, double y, double r, double z){
  grayStuare(x, y, r);

  if (r < 0.05) return;

  if (z % 2 == 0) draw(x - r, y - r, r / 2, z - 1); // lower left
  if (z % 1 == 0) draw(x - r, y + r, r / 2, z - 1); // upper left
  else if (r < 1) draw(x + r, y + r, r / 2, z - 1); // upper right
  if (z % 3 == 0) draw(x + r, y - r, r / 2, z - 1); // lower right
}
}
```



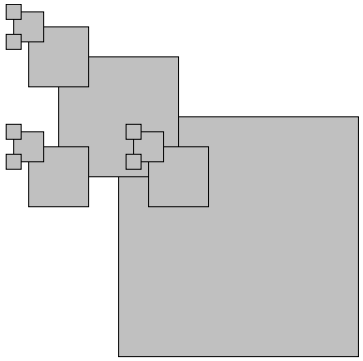
r:0.5 z:3



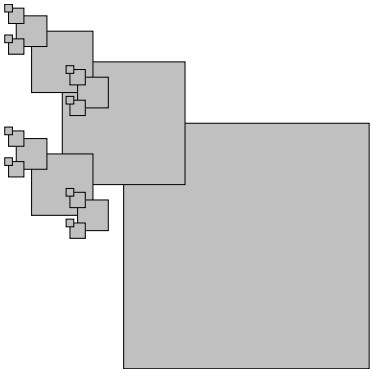
r:0.25 z:2



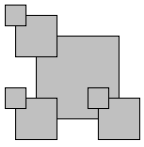
r:0.25 z:3



r:0.5 z:1



r:1 z:5



r:0.125 z:0

Prime Directive

4. (20 points) Using the Allen Telescope Array, the SETI ("Search for Extra-Terrestrial Intelligence") Institute recently picked up an unusual radio signal originating in the Orion nebula. Detailed, top-secret analysis of the signal has revealed it to be a Java program, giving indisputable proof of the language's universal appeal. Unfortunately, nobody has been able to figure out yet what the program does:

```
public class Vulcan {
  public static void main(String[] romulan) {
    int klingon = Integer.parseInt(romulan[0]);
    if (klingon < 1) return;

    for (int ferengi = 2; ferengi <= klingon; ferengi++) {
      if (klingon % ferengi == 0) {
        while (klingon % ferengi == 0) klingon /= ferengi;
        System.out.println(ferengi);
      }
    }
  }
}
```

(a) What does the program `Vulcan` print out when run with the argument "0"? Nothing

(b) What does `Vulcan` print out when run with the argument "1"? Nothing

(c) What does `Vulcan` print out when run with the argument "3"? 3

(d) What does `Vulcan` print out when run with the argument "8"? 2

(e) What does `Vulcan` print out when run with the argument "24"? 2
3

(f) What does `Vulcan` do? Your answer should be at most 30 words.
Print all prime factors of its argument, or nothing if the argument is less than 2.

The Sum of its Parts

5. (30 points) Write two versions of a static function, `sumDigits`, that takes an integer and returns the sum of its digits. For example, the sum of the digits in 1234 is 10 because $1+2+3+4 = 10$, and the sum of the digits in -52 is 7 (the minus sign is not a digit, after all).

The first version, **on this page**, should be iterative and must not make any function calls (not even to library functions like `Math.max()`). The second version, **on the next page**, should be recursive but must not call any other functions and must not contain any loops. You do not need to include comments in either version, and you do not need to write a class around them.

Hint: Start with the version of the function that is most intuitive to you.

(a) Iterative version (no function calls):

This part was graded out of 14 points, with a bonus point if the program was both correct AND elegant.

```
public static int sumDigits(int x) {
    if (x < 0) x = -x;
    int sum = 0;
    while (x > 0) {
        sum += x % 10;
        x /= 10;
    }
    return sum;
}
```

(b) Recursive version (no loops):

This part was graded out of 13 points, with two bonus points if the program was both correct AND elegant.

```
public static int sumDigits(int x) {
    if (x <  0) return sumDigits(-x);
    if (x == 0) return 0;
    return x % 10 + sumDigits(x / 10);
}
```