CIS 110-002 Fall 2012 Midterm, 7 June 2012, Answer Key

Miscellaneous

1. (1 points)

   (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

Money, money, money, money, money ...

2. (4 points) Consider the following recursive function:

```
public static double compoundInterest(double balance, int months, double rate) {
  if (months <= 0) return balance;
  balance = balance * rate;
  return compoundInterest(balance, months - 1, rate);
}
```

   (a) Is this function tail recursive (circle your answer)? YES

   (b) How many times will `compoundInterest(0, 15, 1.1)` call itself recursively? 15 times

   (c) What value will `compoundInterest(0, 15, 1.1)` return? 0

   (d) Will the following non-recursive function produce the same or different results from the recursive function above? Circle your answer. SAME

```
public static double iterativeInterest(double balance, int months, double rate) {
  for (int i = months; i > 0; i--)
    balance = balance * rate;
  return balance;
}
```

Cabin

3. (7 points) Given the following class, answer the questions below and on the next page:

```
public class Fever {
  public static int restless(int[] arr) {
    int sum = 0;
    for (int i = 0; i < arr.length; i++)
      sum = sum + arr[i];
    return sum / arr.length;
  }

  public static void main(String[] args) {
    int[] intArgs = new int[args.length];
    for (int i = 0; i < args.length; i++)
      intArgs[i] = Integer.parseInt(args[i]);
    System.out.println(restless(intArgs));
  }
}
```

(a) Circle the statements below that are true

i. will not compile because the `for` loops do not have curly braces;

ii. will not compile because `restless()` is called inside `System.out.println`;

iii. may crash with an `ArrayIndexOutOfBounds` error when run;

iv. `intArgs[]` is `args[]` converted into an integer array; X

v. prints the approximate average of the command line arguments. X

vi. does not print anything because the `intArgs[]` array is different from the `arr[]` array ;

(b) Assuming that any errors you identified above are corrected, what will the program print out in each of the following cases? We will consider your answer correct as long as it shows you understand what the program will print out.

i.
```
% java Fever
java.lang.ArithmeticException:  / by zero
```

ii.
```
% java Fever 1.3 2.5 7.1
java.lang.NumberFormatException:  For input string:  "1.3"
```

iii.
```
% java Fever 2 4 7
4
```

iv.
```
% java Fever hello world
java.lang.NumberFormatException:  For input string:  "1.3"
```

Find the Bugs

4. (8 points) Identify eight errors in the following program that prevent it from compiling or from running and show how to correct them. Write your answers in the space provided on the next page. The line numbers are for your convenience and are not part of the program.

```
 1: public class Foo() {
 2:   public static int main(String[] args) {
 3:     int rows = args[0];
 4:     drawSomething(rows);
 5:     double hyp = ComputeSomething(rows);
 6:     System.out.println(length(hypotenuse) =" + hyp);
 7:   }

 8:   public static void drawSomething(int rows) {
 9:     for (i = 0; i < rows; i = i++) {
10:       for (j = 0; j < i + 1; j = j + 1)
11:         System.out.print("*");
12:       System.out.println();
13:     }
14:   }

15:   public static double computeSomething(int leg) {
16:     int sqrt2 = Math.sqrt(2);
17:     return sqrt2 * leg;
18:   }
19: }
```

Bug 1: 1: `class Foo {`
Bug 2: 2: `public static void main(String[] args)`
Bug 3: 3: `int rows = Integer.parseInt(args[0])`
Bug 4: 5: `double hyp - computeSomething(rows)` // lower-case c
Bug 5: 6: `System.out.println("length(hypotenuse) = " + hyp);`
Bug 6: 9: `for (int i = 0; i < rows; i++) {`
Bug 7: 10: `for (int j = 0; j < i + 1; j = j + 1)`
Bug 8: 16: `double sqrt2 = Math.sqrt(2);`

On line 9, the increment statement `i = i++` should have read simply `i++`. Although `i = i++` actually works (!), we gave you credit if you identified this as one of the eight bugs. Even though it works, we haven't discussed that sort of statement in class, it is unbelievably bad style, and you were quite right to flag it.

Partial Sums

5. (15 points) The median, or middle value, of a list of numbers is extremely useful in a variety of computer and statistical algorithms. But it is notoriously slow to compute relative to the number of times it needs to be computed. Often, it is better to settle for an approximation of the median than to calculate it exactly. One of the simplest approximations is to pick three random elements, and calculate the median of those three.

Write a function `threeMedian` that takes a single argument `N`, reads in `N` `double`s from standard input using `StdIn.readDouble()`, and returns the median of three randomly chosen elements. Assume that `N` is always at least three, and that all calls to `StdIn.readDouble()` succeed without error. Use `Math.random()` to pick the three random elements (recall that `Math.random()` returns a random number between 0 and 1, but never exactly 1). Do not write the code for the class that contains `threeMedian`, only write the function itself. You do not need to comment your code.

We awarded 14 points on this problem for a completely correct answer, and 15 for a correct *and* elegant answer along the lines of the second solution below.

```
public static double threeMedian(int N) {
    double[] arr = new double[N];

    for (int i = 0; i < N; i++)
        arr[i] = StdIn.readDouble();

    double a = arr[(int) (N * Math.random())];
    double b = arr[(int) (N * Math.random())];
    double c = arr[(int) (N * Math.random())];

    if (a <= b && b <= c) return b;
    else if (c <= b && b <= a) return b;
    else if (b <= a && a <= c) return a;
    else if (c <= a && a <= b) return a;
    else return c;
}

// ---------- or ----------

public static double threeMedian(int N) {
    double[] arr = new double[N];

    for (int i = 0; i < N; i++)
        arr[i] = StdIn.readDouble();

    double a = arr[(int) (N * Math.random())];
    double b = arr[(int) (N * Math.random())];
    double c = arr[(int) (N * Math.random())];

    double min = Math.min(Math.min(a, b), c);
    double max = Math.max(Math.max(a, b), c);
    return a + b + c - min - max;
}
```

Tracery

6. (15 points)

For each of the labeled points in the code fragment below, identify each of the assertions in the table as being *sometimes*, *always*, or *never* true. Assume that `bar` is only called from within `foo`, and that the values of all `int`s stay within the valid range for integers (i.e. no value will grow so large that it will wrap around become negative, or vice cersa).

Abreviate sometimes with **S**, always with **A**, and never with **N**.

```
public static void foo(int a, int b, int c) {
  if (a > b && b <= 0)
    b = Math.abs(b - a) + 1;
  else if (a < b)
    c = Math.abs(c) + 1;
  else
    a = b - 1;

  // Point A

  if (c > a) {
    c = a;
    a = b * b + 2;
    // Point B
  } else {
    c = a;
    c = bar(b, Math.abs(a), c * c);
    // Point C
  }

  if (b == a)
    b--;

  // Point D
}

public static int bar(int c, int a, int b) {
  for (int i = 0; i < a; i++)
    b = b - a;

  // Point E

  return c + b;
}
```

	a <= 0	c <= b	b >= a
A	S	S	A
B	N	A	N
C	S	A	A
D	S	A	S
E	S	S	S