CIS 110 Spring 2012 Final Exam, 7 May 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

Find the Bugs

2. (5 points) Give five bugs in the following code. The line numbers are included for your convenience and are not part of the program. (Hint: Think about what the program is trying to do to find bugs in the logic.)

```
1:  int[][] myarray = int[5][4];
2:  int sum;
3:  for (int j = 0; j < myarray.length; j++) {
4:      for (int i = 0; j < myarray[0].length; i++) {
5:          myarray[i][j] = i + j;
6:          int sum = i + j;
7:      }
8:  }
9:  System.out.println(sum);
```

Bug 1: `1: int[][] myarray = new int[5][4];`
Bug 2: `2: int sum = 0;`
Bug 3: `4: i < myarray[0].length`
Bug 4: `5: myarray[j][i] = i + j;`
Bug 5: `6: sum += i + j;`

Method Madness

3. (9 points) Consider the following program:

```
public class Madness {
    public static int[] y = { 7, 3, 2 };
    public static int   x = 2;

    public static void main(String[] args) {
        int x = Integer.parseInt(args[0]);
        x = mysterymethod(x, y);
        System.out.println(x);
        for (int i = 0; i < y.length; i++) {
            System.out.println(y[i]);
        }
    }

    public static int mysterymethod(int y, int[] z) {
        for (int i = 0; i < z.length; i++)
            z[i] += x;
        return x - y;
    }
}
```

(a) What will "`java Madness 7`" print out?

```
-5
9
5
4
```

(b) What will "`java Madness 3`" print out?

```
-1
9
5
4
```

(c) What will "`java Madness 2`" print out?

```
0
9
5
4
```

Why

4. (12 points) Consider the following class, then answer the questions on the following page:

What will each of the following code snippets print? If the code will cause a runtime error, write `Exception`.

(a)
```
Why w = new Why();
System.out.println(w);
```
Exception

(b)
```
Why w = new Why("Why did the chicken cross the road?");
System.out.println(w);
```
Why Why did the chicken cross the road??
Why not?

(c)
```
Why w = new Why("Why did the chicken cross the road?", "To get to the other side.");
System.out.println(w);
```
Why Why did the chicken cross the road??
To get to the other side.

(d)
```
Why w = new Why(null, "");
System.out.println(w);
```
Exception

(e)
```
Why w = new Why("", null);
System.out.println(w);
```
Why?
null

(f)
```
Why w = new Why(null, null);
System.out.println(w);
```
Exception

TOY

5. (13 points) Consider what happens when the following TOY program is executed by pressing RUN with the program counter set to 10:

```
01: 0001   ( 0000 0000 0000 0001 )
02: 0010   ( 0000 0000 0001 0000 )
03: 0012   ( 0000 0000 0001 0002 )
10: 8101   R[1] <- Mem[01]
11: C014   if (R[0] == 0) pc <- 14
12: 1331   R[3] <- R[3] + R[1]
13: 9302   Mem[02] <- R[3]
14: 8202   R[2] <- Mem[02]
15: 2321   R[3] <- R[2] - R[1]
16: C320   if (R[3] == 0) pc <- 20
17: 91FF   mem[FF] <- R[1]
18: 1111   R[1] <- R[1] + R[1]
19: C014   if (R[0] == 0) pc <- 14
20: 0000   halt
```

(a) Describe, in 20 words or less, what the above program does:
The program prints out 1 2 4 8 (hex).

(b) Describe, in 20 words or less, what the program would do if the instruction at memory address 11 were replaced with the following:

```
11: 8302   R[3] <- Mem[02]
```

The program prints out powers of 2 until it wraps around, then prints zeros forever.

Whiteboarding

6. (20 points) The list of names of students waiting for help during office hours is a good example of a ring buffer. Students add their names to the end of the list, and TAs help students from the front of the list. When the list reaches the end of the board, it wraps around. But there is one wrinkle: students only get to have their name on the board at most once. Consider the following class to implement this behavior, then answer the two questions on the following pages:

(a) Implement the `enqueue()` method so that it behaves according to its comment. Make sure the values of all instance variables are consistent with their comments when `enqueue()` returns. (Hint: recall that if `s` and `t` are strings, then `s.compareTo(t)` will return `0` if the two strings are the same.)

```
public void enqueue(String name) {
    if (name == null) return;
    if (isFull()) return;

    for (int i = first; i != last; i = (i + 1) % rb.length)
        if (name.compareTo(rb[i]) == 0) return;

    rb[last] = name;
    last = (last + 1) % rb.length;
    size++;
}
```

(b) What will the contents of the `rb` array be after the following sequence of statements? Indicate which entries in the array correspond to `first` and `last`

Your answer:

```
Jeff
Nate
Catherine  <- first, last
Rupi
Kathryn
```

Binary Search Trees

7. (30 points) A binary search tree (BST) is a linked data structure, where each node contains a value and pointers to two other nodes, `left` and `right`. These nodes are referred to as its *children*. Just as the `next` node of a linked list is itself the start of a linked list, the `left` and `right` children of a BST node are themselves BSTs, called the left and right *sub-trees*. What makes BSTs interesting is an additional requirement that the every value in the left sub-tree of a node `n` must be less than the value stored in `n`, and every value stored in the right sub-tree of `n` must be greater than or equal to the value stored in `n`. If the left or right sub-trees of `n` are empty, then the `left` and/or `right` pointers will be `null`. Here are some examples of BSTs:

(a) Implement the recursive `insert()` method as described in the problem description.

```
public void insert(Comparable v) {
    if (v.compareTo(value) < 0) {
        if (left != null) left.insert(v);
        else left = new BST(v);
    } else {
        if (right != null) right.insert(v);
        else right = new BST(v);
    }
}
```

(b) Implement the recursive `printReverseSorted()` method to print all the values in the tree in reversesorted order, as described in the problem description.

```
public void printReverseSorted() {
    if (left != null)  right.printReverseSorted();
    printValue();
    if (right != null) left.printReverseSorted();
}
```