CIS 110 Fall 2012 Final, 17 December 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

True/False

2. (5 points)
   For each question below, circle the correct answer:

   (a) ~~TRUE~~ FALSE For an array `arr`, `arr[length]` will return the last item in the array.

   (b) ~~TRUE~~ FALSE Java classes cannot mix static and non-static methods and variables.

   (c) TRUE ~~FALSE~~ One array cannot contain both `int` and *double* elements.

   (d) ~~TRUE~~ FALSE `(int) (6 * Math.random())` will return a random integer between 1 and 6.

   (e) ~~TRUE~~ FALSE `java printOutput | out.txt` will redirect the output of `printOutput` to the file `out.txt`

   (f) TRUE ~~FALSE~~ The command "`cd ..`" will move to the parent directory in the terminal or command prompt.

   (g) TRUE ~~FALSE~~ `0x001A` in hexadecimal is the same as `00011010` in binary.

   (h) ~~TRUE~~ FALSE In two's complement notation, a binary number has a 1 in the highest-order (most significant) bit if it is positive.

   (i) ~~TRUE~~ FALSE A constructor is the only type of function that can be overloaded.

   (j) TRUE ~~FALSE~~ In a linked list, elements do not have to be sequential in memory.

Babes in TOYland

3. (15 points) The following TOY program does something, but we've forgotten what. All we remember is that the program reads a single value from standard input (at memory address 0x11), and writes a single value to standard output (at memory address 0x21). Answer the questions below, then remind us what the program does. You may assume that the assembly language comments are correct and that the number read from standard input is not negative. Give numeric answers below in ordinary, boring, base 10.

```
01: 0001   (0000 0000 0000 0001,      1)

10: 8501   R[5] <- mem[01]
11: 83FF   read R[3]
12: 1455   R[4] <- R[5] + R[5]
13: 2230   R[2] <- R[3]
14: 1102   R[1] <- R[2]
15: B004   mem[R[4]] <- R[0]
16: C120   if (R[1] == 0) goto 20
17: C21D   if (R[2] == 0) goto 1D
18: AA04   R[A] <- mem[R[4]]
19: 1AA3   R[A] <- R[A] + R[3]
1A: BA04   mem[R[4]] <- R[A]
1B: 2225   R[2] <- R[2] - R[5]
1C: C017   goto 17
1D: 2115   R[1] <- R[1] - R[5]
1E: 1203   R[2] <- R[3]
1F: C016   goto 16
20: 8B02   R[B] <- mem[02]
21: 9BFF   write R[B]
22: 0000   halt
```

(a) What is the value in register `R[4]` at memory address 0x15? 2

(b) What value does the program print if it reads zero from standard input? 0

(c) What value does the program print if it reads one? 1

(d) What value does the program print if it reads five? 125

(e) Describe, **in twenty words or less**, what this program does.

The program prints the cube of its input.

Unbreakable

4. (20 points) For each of the four functions below and on the next page, give an intuitive description **in 20 words or less** of the function's purpose. In addition, **in 30 words or less** state whether the function will compile and run properly in all cases and what the error(s) will be otherwise. Circle your answer.

(a)
```
public static void one(int x) {
  while (x != 0) {
    System.out.println(x);
    x--;
  }
}
```

Prints all integers from `x` down to 1. If $x < 0$, wraps around from -2^{31} to $2^{31} - 1$. *We accepted any answer that recognizes the wrap-around, or that states the program enters an infinite loop if* $x < 0$

(b)
```
public static int two(char x) {
  String keyboard = "abcdefghijklmnopqrstuvwxyz";
  for (int i = 0; i < keyboard.length(); i++)
    if (keyboard.charAt(i) == x)
      return i;
}
```

Returns the letter of the alphabet that was typed. Will not compile, because there is no return statement to handle the case where `x` is not a lower-case letter.

(c)
```
public static int three(int[] x) {
  if (x == null) return 0;

  int sum = 0;
  for (int i = 0; i < x.length; i++)
    sum += x[i];
  return sum / x.length;
}
```

Returns the average of all entries in `x`, truncated to an integer. Gives an `ArithmeticException: / by zero` if `x.length == 0`

(d)
```
public static void four(String[] x) {
  for (int i = 0; i < x.length; i++) {
    x[i] = "" + x[i].length();
    System.out.println(x[i]);
  }
}
```

Prints the lengths of each string in `x`. Gives a `NullPointerException` if `x == null` or if any entry in `x` is `null`.

Something or Other

5. (25 points) Consider the following program, then answer the questions on the next page: What does the program `Something` print when run with the following arguments?

(a) `% java Something 0`

```
sum:  0
foo:  1
```

(b) `% java Something 1`

```
sum:  1
foo:  3
```

(c) `% java Something 2`

```
sum:  4
foo:  7
```

(d) `% java Something 3`

```
sum:  5
foo:  7
```

(e) `% java Something 4`

```
sum: 12
foo: 15
```

Know Your Node Code

6. (30 points) Consider a linked list of integer values that uses the linked list node type:

```
public class Node {
  public Node next;
  public int  value;
}
```

Write a `public static` function `remove` that takes a linked list of `Nodes` `list` and an integer `val` as arguments, modifies `list` by removing all nodes with the value `val`, and returns the modified list. You only need to write the `remove()` function, not the surrounding class. (Hint 1: You do not need to create any new nodes with the `new` operator. Hint 2: Don't look for a clever solution, just write something that works.)

```
/***** ITERATIVE VERSION *****/
public static Node remove(Node list, int val) {
    // remove all instances at the head of the list
    while (list != null && list.value == val)
        list = list.next;

    // check for empty list
    if (list == null) return list;

    // now the head of the list definitely has a value different from val

    // each iteration, either remove an element or go to the next element
    //
    // there is no update in the loop because we do different
    // things depending on whether we remove an element or advance
    for (Node n = list; n.next != null; ) {
        if (n.next.value == val) n.next = n.next.next;
        else n = n.next;
    }

    return list;
}

/***** RECURSIVE VERSION *****/
public static Node remove(Node list, int val) {
    if (list == null) return null;           // base case: empty list
    if (list.value == val) {
        return remove(list.next, val);       // remove from head of list
    } else {
        list.next = remove(list.next, val); // remove from rest of list
        return list;
    }
}
```