COS 126	General Computer Science	Fall 2016

Written Exam 1

0. **Miscellaneous.**

   Don't forget to write your name, NetID, precept, and exam room.

1. **Java basics.**

Java expression	*value*
`x`	`1`
`1.0 * x + y / z`	`1.0`
`((x > y) || (y > z)) && (z > x)`	`false`
`Math.min(Math.max(x, y), y*y % z)`	`1`
`Math.sqrt(x - y - z)`	`NaN` or `Double.NaN`
`Double.parseDouble(x + "2" + y)`	`122.0`

2. **Properties of arrays and functions.**

   (a) T T F F F

   (b) T T F F T

3. **Debugging and arrays.**

   D F G A C

4. **Functions.**

(a) CKDKEK

The letters C, D, and E can be permuted in any order. EKGKHK is an alternative solution, but poorer style.

```
public static boolean oddParity(boolean x, boolean y, boolean z) {
    int count = 0;
    if (x) count++;
    if (y) count++;
    if (z) count++;
    return (count % 2) != 0;
}
```

(b) GEHFE

GEHFJ, GIHFE, GIHFJ are alternative solutions, but poorer style.

```
public static boolean oddParity(boolean x, boolean y, boolean z) {
    if      (x && y) return  z;
    else if (x || y) return !z;
    else             return  z;
}
```

This part is tricker than it might appear. It is tempting to start with IAGB, but then you quickly get stuck.

5. **Recursion.**

(a)

n	0	1	2	3
f(n)	"0"	"1"	"1201"	"1201311201"

(b) T T T

- Let L_n be the length of the return value of `f(n)`. Then, $L_n = L_{n-1}+1+L_{n-2}+L_{n-1}$, with $L_0 = L_1 = 1$. From above, $L_2 = 4$ and $L_3 = 10$. Thus, $L_4 = 10+1+4+10 = 25$ and $L_5 = 25 + 1 + 10 + 25 = 61$.
- Exchanging statements 2 and 3 can have an effect only when `n` is `1`. When `n` is `1`, the modified function still returns `"1"`—it just performs the unnecessary work of calling `f(0)` before doing so.
- The function `f()` has no side effects (such as printing to standard output). Thus `first = f(n-1)` and `third = f(n-1)` will always be equal, and the return value in the original function (`first + n + second + third`) will always equal the return value in the modified function (`first + n + second + first`).

6. **Powers of 2.**

   D B F D C N E E

7. **TOY.**

   The TOY program computes the smallest power of 2 that is strictly greater than a given integer.

```
10: 8A00    R[A] <- M[00]              load a from M[00]
11: 7101    R[1] <- 1                  power = 1
12: 221A    R[2] <- R[1] - R[A]        while (power <= a)
13: D216    if (R[2] > 0) PC <- 16     {
14: 1111    R[1] <- R[1] + R[1]            power = 2*power
15: C012    goto 12                    }
16: 9101    M[01] <- R[1]              store power to M[01]
17: 0000    halt
```

   (a) `0002`

   (b) `0008`

   (c) `0010`

   Remember that everything is in hex: $10_{16} = 16_{10}$.

   (d) `2000`

   You will get this part only by reasoning about what the TOY program does (or wasting an extraordinary amount of time tracing code).

   (e) `0001`

   `FACE` *is a negative integer (two's complement representation). Thus, the loop is skipped and* `R[1]` *remains* `0001`.