COS 126	**General Computer Science**	**Fall 2003**

Exam 1

1. **Number systems.**

   (a) `1111 0011 0111 0110 = F376`.

   (b) `C99A`

   Repeadely dividing by 16 and reading the remainders backwards isn't too painful with the given integer.

2. **Data types.**

   (a) 2.0

   Thinking you had to take the square root of 33/7 without a calculator should have tipped you off to integer division.

   (b) true

   (c) 12

   The `for` loops repeatedly (i) checks the loop continuation condition, (ii) executes the body of the loop, and (iii) does the increment statement.

   (d) 010111010

   String concatenation.

3. **Debugging.**

   b, g, f, i, a

4. **Loops and conditionals.**

   (a)
   ```
   A . B
   . A .
   B . A
   ```

   (b)
   ```
   A . . . B
   . A . B .
   . . A . .
   . B . A .
   B . . . A
   ```

5. **Input, output, loops, arrays, debugging.**

   The program fills up an array of size N with the first N strings in the text file. Then it prints out every 3rd string (with wrap-around).

(a)
```
thieves
scum
buckle
goes
lowlives
that
```

(b) infinite loop

```
thieves
scum
thieves
scum
...
```

(c) `java.lang.ArrayIndexOutOfBoundsException`

The 6 words that `java Lyrics 10` outputs are the input to `java Lyrics 2`. However, the array in the second program only has 2 elements so accessing `a[x]` when x is 3 will be out-of-bounds.

6. **Using arrays.**

a, d, e

7. **Functions.**

```
static int min6(int a, int b, int c, int d, int e, int f) {
    int x = min3(a, b, c);
    int y = min3(d, e, f);
    return  min3(x, x, y);
}
```

Or, if you don't mind calling the Java library function `Math.min`.

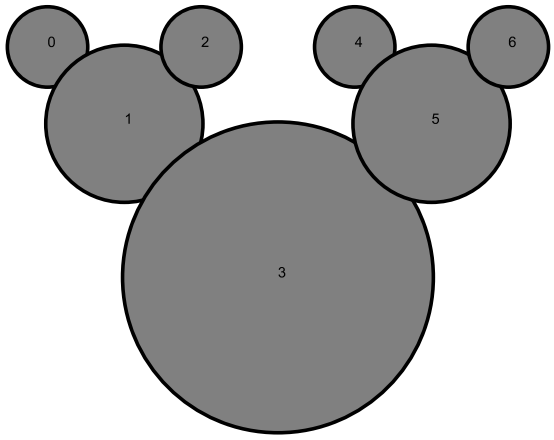
```
static int min6(int a, int b, int c, int d, int e, int f) {
    return Math.min(min3(a, b, c), min3(d, e, f));
}
```

Or, for a more obfuscated solution that doesn't call `min3()`, but satisfies the letter, if not spirit, of the question.

```
static int min6(int a, int b, int c, int d, int e, int f) {
    while (d <= a && d <= b && d <= c && d <= e && d <= f)
        return d;
    return min6(b, d, a, e, f, c);
}
```

8. **Recursion.**

This recursive function is identical to the one on the H-tree assignment, except that it draws a circle instead of an H, and doesn't make the bottom left or bottom right recursive calls. Also, the `drawCircle` command is in between the two recursive calls. This determines the order in which the calls are made (but not the pattern itself).



9. **TOY I.**

   The program loads `AAAA` into register A and `BBBB` into register B. It loads the contents of memory cell 02 into register C. If this value is 0, then it writes register A to memory cell 01; otherwise it writes register B to memory cell 00. It's an `if-else` statement in TOY.

```
00: AAAA
01: BBBB
02: 0000

10: 8A00   RA <- mem[00]
11: 8B01   RB <- mem[01]
12: 8C02   RC <- mem[02]
13: CC16   if (RC == 0) goto 16
14: 9A01   mem[01] <- RA
15: C017   goto 17
16: 9B00   mem[00] <- RB
17: 0000
```

   (a) `BBBB BBBB 0000`

   (b) `AAAA AAAA 0005`

10. **TOY II.**

   (a) `0000`

   The program repeatedly reads integers from standard input and XOR them together. It terminates when it reads the value `0000`. Recall that XORing a bit with itself always yields 0. Thus, `a ^ a = 0` for any integer `a`.

   (b) `ACDC`

   Observe that the XOR of a sequence of integers is independent of the order in which you do it. That is `a ^ b ^ a ^ b = a ^ a ^ b ^ b = 0`. Thus, all the integers cancel each other out except `ACDC`. a bunch of integers together