COS 126	General Computer Science	Fall 2004

Exam 1

This test has 10 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Signature

Problem	Score
0	
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
9	
Sub 2	

Total	

Name:

NetID:

Precept:

1	MF 10:00	Donna
2	MF 11:00	Eddie
3	MF 1:30	Chris B.
4	MF 2:30	Kevin
5	M 7:30 F 2:30	Chris D.

0. **Miscellaneous. (2 points)**

   (a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle your precept number.

   (b) *Write* and sign the honor code on the front of the exam.

1. **Number systems. (4 points)**

   (a) What is the decimal representation of the binary integer 110110_2? Circle your answer.

   (b) Convert the decimal integer 111_{10} to binary. Circle your answer.

   (c) What is the sum of the two binary integers 110110_2 and 1110101_2? Give the answer in hex. Circle your answer.

   (d) Suppose that `a` and `b` are Java variables of type `int` (32-bit two's complement integers). Find values for `a` and `b` for which the following condition evaluates to `true`.

```
(a > b) && (a - b < 0)
```

2. **Debugging. (5 points)**

The following program reads in a sequence of integer values between 0 and 99 inclusive, and prints out the the value that occurs most often (the *mode*). The program compiles, but there are some bugs which prevent it from calculating the correct answer. The line numbers are for reference.

```
 1  public class Mode {
 2      public static void main (String[] args) {
 3
 4          int i = 0;
 5          int maxi = 0;
 6          int[] a = new int[99];
 7
 8          while(!StdIn.isEmpty())
 9              i = StdIn.readInt();
10              a[i]++;
11
12          for (i = 0; i <= 99; i++);
13              if (a[i] > a[maxi]) maxi = i;
14
15          System.out.println(maxi + " occurs " + a[maxi] + " times");
16      }
17  }
```

(a) Identify all of the mistakes.

(b) What will the corrected program do if there is more than one number that occurs with the greatest frequency?

3. **Loops and conditionals. (4 points)**

Consider the following program.

```
public class Pattern {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                System.out.print((N - i + j) % N);
            }
            System.out.println();
        }
    }
}
```

What is the result of the following command? Circle your answer.

```
java Pattern 6
```

4. **Java basics. (5 points)**

   Write a complete program `SignalAnalyzer.java` that reads in a sequence of real values between -1 and 1 from standard input and prints out their *average magnitude*. The average magnitude is the average of the absolute values of the inputs. Assume that you have access to the library `StdIn.java`. Your answer will be graded for correctness and clarity.

```
public class SignalAnalyzer {
    public static void main (String[] args) {









    }
}
```

5. **Recursive graphics. (6 points)**

Suppose that the function `drawShadedCircle(x, y, d)` plots a filled-in gray circle with a black outline, of diameter d, centered on (x, y). Consider the following recursive function.

```
public static void recur(double x, double y, int depth, double size) {
1   if (depth <= 0) return;
2   drawShadedCircle(x, y, size);
3   recur(x,          y + size/4, depth-1, size/2);
4   recur(x + size/4, y,          depth-1, size/2);
5   recur(x,          y - size/4, depth-1, size/2);
6   recur(x - size/4, y,          depth-1, size/2);
}
```

For each of the following orderings of the six statements, give the picture that results when you call `recur(256, 256, 2, 512)`.

(a) `1 2 3 4 5 6` ☐

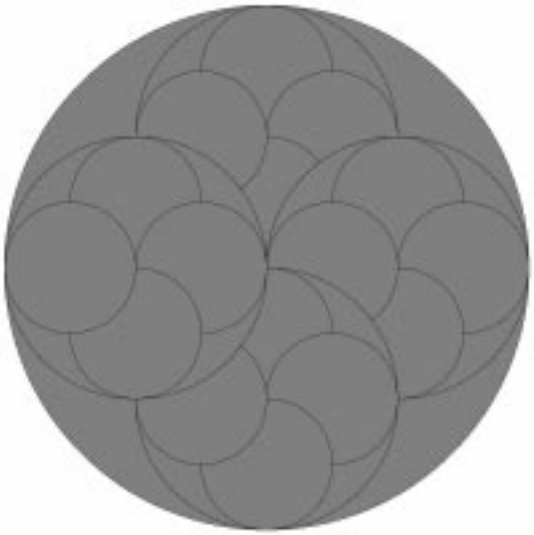
(b) `1 3 6 2 5 4` ☐

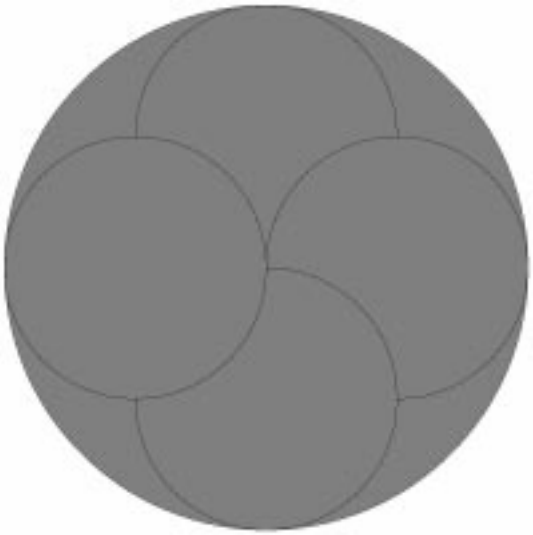
(c) `1 6 3 4 5 2` ☐

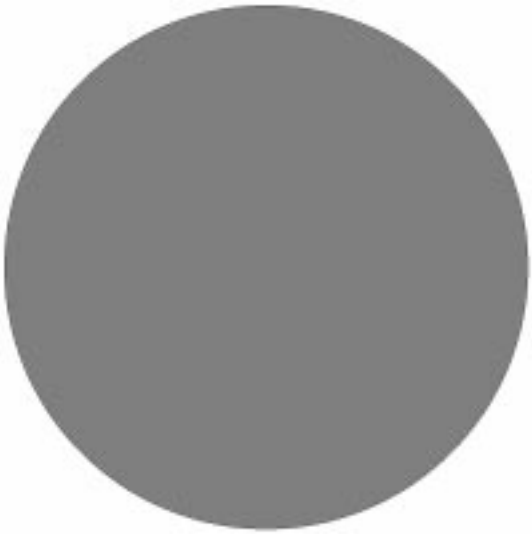
(d) `2 1 3 4 5 6` ☐

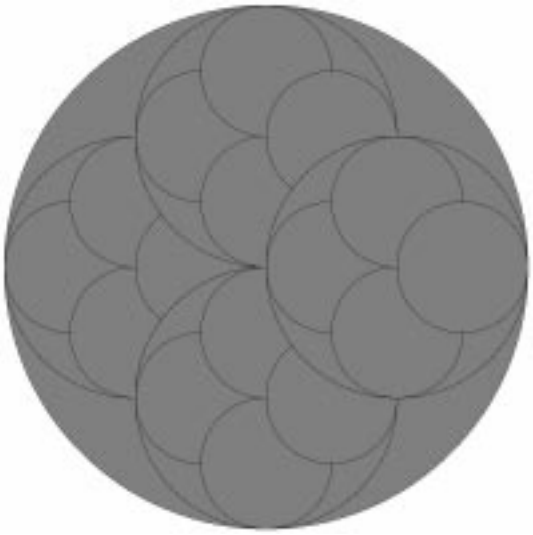
(e) `2 1 6 3 5 4` ☐

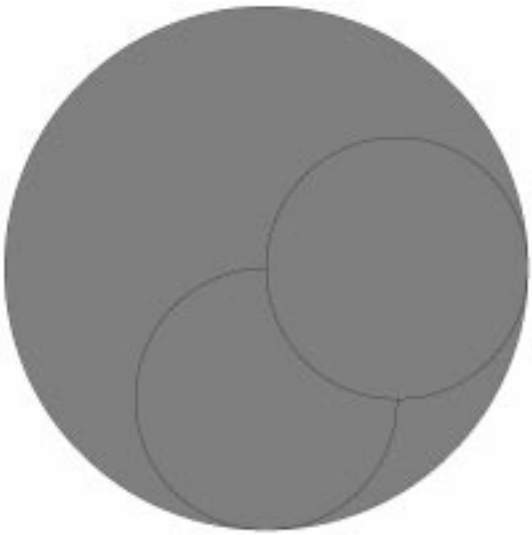
(f) `2 3 4 5 6 1` ☐

(i) 

(ii) 

(iii) 

(iv) 

(v) 

(vi) `java.lang.StackOverflowError`

6. **TOY. (7 points)**

Suppose that you load the following into locations `10`–`14` and `F0`–`F6` of TOY, set the program counter to `10`, and press RUN.

```
10: 7100  R1 <- 00          F0: AA01  RA <- mem[R1]
11: 7201  R2 <- 01          F1: AB02  RB <- mem[R2]
12: 7F14  RF <- 14          F2: 2CBA  RC <- RB - RA
13: C0F0  pc <- F0          F3: DCF6  if (RC > 0) pc <- F6
14: 0000  halt              F4: BB01  mem[R1] <- RB
                            F5: BA02  mem[R2] <- RA
                            F6: EF00  pc <- RF
```

(a) Suppose that you set memory locations `00` and `01` as follows:

```
00: 00BE
01: 0060
```

What are the values stored in memory locations `00` and `01` when the program terminates?

```
00:
01:
```

(b) Explain briefly what the above program does to the two positive integers stored initially in memory locations `00` and `01`.

Replace memory locations 14–1C with the following values, and keep 10–13 and F0–F6 the same as in part (a).

```
10: 7100  R1 <- 00
11: 7201  R2 <- 01
12: 7F14  RF <- 14
13: C0F0  pc <- F0

14: 7101  R1 <- 01
15: 7202  R2 <- 02
16: 7F18  RF <- 18
17: C0F0  pc <- F0

18: 7100  R1 <- 00
19: 7201  R2 <- 01
1A: 7F1C  RF <- 1C
1B: C0F0  pc <- F0

1C: 0000  halt
```

(c) Suppose that you set memory locations 00–02 as follows:

```
00: 00BE
01: 0060
02: 000D
```

What are the values stored in memory locations 00–02 when the program terminates?

```
00:
01:
02:
```

(d) Explain briefly what the above program does to the three positive integers stored initially in memory locations 00–02.

7. **Functions. (4 points)**

   Write a function `majority` that takes three `boolean` inputs and returns `true` if at least two of the inputs are `true`, and `false` otherwise. You may **not** use an `if` statement. Your answer will be graded for correctness and clarity.

8. **Arrays. (8 points)**

   For the purposes of this question, a *ranking* of `N` elements is an integer array of size `N`, containing each of the integers from 0 through N-1 exactly once.

   Given a ranking `a[]` of size `N`, the *inverse ranking* `ainv[]` is an array of size `N` such that for each i, we have `ainv[a[i]] = a[ainv[i]] = i`. For example:

```
a:     0 5 1 2 3 4        b:     1 2 0 5 3 4
ainv:  0 2 3 4 5 1        binv:
```

   (a) Write down the inverse ranking `binv` in the space above.

   (b) Write a Java fragment that creates and initializes the inverse ranking array `binv`, given `b` and `N`.

(c) Given `ainv`, write a *one-line* Java fragment that evaluates to `true` if i appears before j in the ranking `a`, and `false` otherwise. In the example above, 3 appears before 4, but 3 does not appear before 1.

(d) The *Kendall tau distance* between two rankings `a` and `b` is the number of pairs of integers that appear in opposite order in the two rankings. For example, if the two ranking arrays are `a` and `b` as below,

```
a:  0 5 1 2 3 4
b:  1 2 0 5 3 4
```

then the Kendall tau distance is 4 since the the following four pairs disagree: 0-1, 0-2, 1-5, 2-5. We count 0-1 as a disagreement since 0 appears before 1 in `a` but not in `b`; we count 1-5 since 1 appears after 5 in `a` but not in `b`.

Given two ranking arrays `a` and `b`, each of size `N`, write a Java code fragment to compute their Kendall tau distance. You may assume that, in addition to `a`, `b`, and `N`, you have access to `ainv` or `binv`, as defined in part (a). Your answer will be graded for correctness and the clarity of the supporting comments that explain what your code is doing.

9. **Input, output. (5 points)**

```
public class Conway {
    public static void main(String[] args) {
           int current = StdIn.readInt();
           int inarow = 1;
           while(!StdIn.isEmpty()) {
              int next = StdIn.readInt();
              if (next != current) {
                  System.out.print(inarow + " " + current + " ");
                  inarow  = 0;
                  current = next;
              }
              inarow++;
           }
           System.out.println(inarow + " " + current);
      }
  }
```

Suppose that the file `input.txt` contains the following sequence of integers.

```
1 1 1 2 2 2 3 3 3 3 6 6 6 1 1 1 1 1 1
```

What happens when you execute the program above with the following commands? Circle your answers.

(a) `java Conway < input.txt`

(b) `java Conway < input.txt > output.txt`

`java Conway < output.txt`

(c) `java Conway < input.txt | java Conway | java Conway`

This page intentionally left blank.

TOY REFERENCE CARD

INSTRUCTION FORMATS

```
            | . . . . | . . . . | . . . . | . . . .|
  Format 1: | opcode  |    d    |    s    |    t   |  (0-6, A-B)
  Format 2: | opcode  |    d    |       addr       |  (7-9, C-F)
```

ARITHMETIC and LOGICAL operations

```
    1: add              R[d] <- R[s] +  R[t]
    2: subtract         R[d] <- R[s] -  R[t]
    3: and              R[d] <- R[s] &  R[t]
    4: xor              R[d] <- R[s] ^  R[t]
    5: shift left       R[d] <- R[s] << R[t]
    6: shift right      R[d] <- R[s] >> R[t]
```

TRANSFER between registers and memory

```
    7: load address     R[d] <- addr
    8: load             R[d] <- mem[addr]
    9: store            mem[addr] <- R[d]
    A: load indirect    R[d] <- mem[R[t]]
    B: store indirect   mem[R[t]] <- R[d]
```

CONTROL

```
    0: halt             halt
    C: branch zero      if (R[d] == 0) pc <- addr
    D: branch positive  if (R[d] >  0) pc <- addr
    E: jump register    pc <- R[d]
    F: jump and link    R[d] <- pc; pc <- addr
```

Register 0 always reads 0.
Loads from mem[FF] come from stdin.
Stores to mem[FF] go to stdout.