COS 126	General Computer Science	Spring 2011

Exam 1

This test has 8 questions worth a total of 50 points. You have 50 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."

Name:

`Signature`

NetID:

Problem	Score
0	
1	
2	
3	
4	
5	
6	
7	
Total	

P01	TTh 1:30	Keith
P01A	TTh 1:30	Doug
P01B	TTh 1:30	Victor
P01C	TTh 1:30	Richard
P01D	TTh 1:30	Gordon
P01E	TTh 1:30	Arman
P02	TTh 2:30	Doug
P03	TTh 3:30	Gordon
P03A	TTh 3:30	Keith
P04	TTh 7:30	Nick
P05	WF 10	Dmitry
P06	WF 1:30	Victor
P06A	WF 1:30	Chris
P06B	WF 1:30	Donna
P07	WF 12:30	Donna

Do not remove this exam from the room.

0. **Miscellaneous. (1 point)**

   (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) Suppose that a TOY memory location holds the value `00AD`. What is the corresponding value in decimal? Circle your answer.

   (b) How many values does the following `for` loop print? Recall that a Java `int` is a 32-bit two's complement integer. Circle the correct answer.

```
for (int i = 1; i >= 0; i = i + i) {
    StdOut.println(i);
}
```

0   1   30   31   32 $2^{30} - 1$ $2^{31} - 1$ $2^{32} - 1$ infinite loop

2. **Java basics. (10 points)**

(a) Give the type and value of each of the following Java expressions. If it leads to a compile-time or runtime-error, specify that for the type (and leave the value column blank).

Java expression	*type*	*value*
`1 + 2.0 * 3 + 4.0`		
`(-1 / -1) / 0`		
`(-1.0 / -1.0) / 0.0`		
`Math.sqrt(-2.0)`		
`1 + "+" + 2.0 + "3"`		
`(double) (10 / 4)`		
`(1.0 <= 2.0 <= 3.0)`		

(b) Which of the following are true of Java arrays. Circle all that apply.

i. Array entries are auto-initialized to `0.0` when creating an array of type `double[]`.

ii. Can change the size of the array after creation.

iii. Given an array `a[]` that has been declared and initialized, accessing `a[a.length]` results in a runtime error.

iv. Can use an array as a return type from a function.

v. Can pass an array to a function and have that function change the values stored in the array entries.

3. **Loops, conditionals, and arrays. (8 points)**

Consider the following Java code fragment.

```
int N = a.length;
double min = Double.POSITIVE_INFINITY;
for (int i = 0; i < N; i++) {
    for (int j = i+1; j < N; j++) {
        double delta = Math.abs(a[i] - a[j]);
        if (delta < min) {
            min = delta;
        }
    }
}
```

(a) Suppose that the array `a[]` is initialized as follows

```
double[] a = { 4.5, 3.5, 6.0, 20.0, 3.0 };
```

What is the value of `min` upon termination of the nested `for` loops? Circle your answer.

(b) Given an array `a[]`, describe in *15 words or less* the value of `min` upon termination.

4. **Input and output. (6 points)**

Consider the following Java program.

```
public class Mystery {
    public static void main(String[] args) {
        int curr = StdIn.readInt();
        StdOut.print(curr + " ");
        int prev = curr;

        while (!StdIn.isEmpty()) {
            curr = StdIn.readInt();
            StdOut.print((prev + curr) / 2 + " ");
            prev = curr;
         }
         StdOut.println();
    }
}
```

Assume the contents of the file `input.txt` are given below.

```
% more input.txt
2  4  6  8  10  12  8   2
```

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

```
% java Mystery < input.txt
```

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

```
% java Mystery < input.txt | java Mystery
```

5. **Functions. (8 points)**

The `gcd()` function, defined in a class `Euclid`, takes two nonnegative integer arguments and return the greatest common divisor of the two integers.

```
public class Euclid {
    public static int gcd(int p, int q) {
        if (q == 0) return p;
        return gcd(q, p % q);
    }
}
```

(a) Write an overloaded function `gcd()` that takes *three* nonnegative integer arguments and returns the greatest common divisor of the three integers. Assume that the function is in the same class `Euclid` as the two-argument version above.

Hint: Use the identity $gcd(p,\, q,\, r) \;=\; gcd(gcd(\,p,\, q),\, r)$. For example, $gcd(504,\, 4116,\, 4410) \;=\; gcd(gcd(504,\, 4116),\, 4410) \;=\; gcd(84,\, 4410) \;=\; 42.$

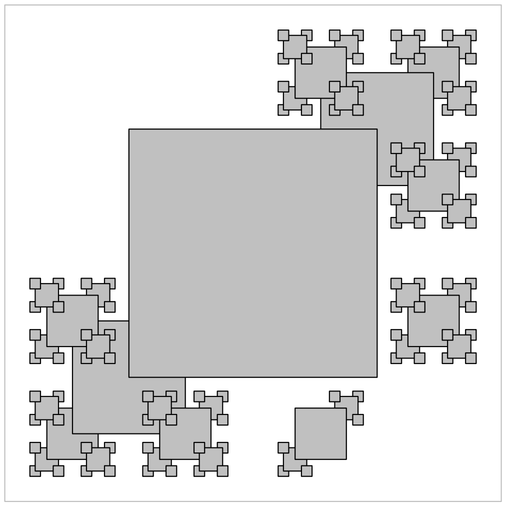
(b) Give the *signature* of a function that takes as an argument an array of nonnegative integers and returns the greatest common divisor of those integers. *Do not implement the function.*

6. **Recursive graphics. (7 points)**

Design a recursive function with the signature

```
public static void draw(int n, double x, double y, double size)
```

so that the call `draw(5, 0.5, 0.5, 0.5)` produces the following *intermediate* result after drawing the 203^{rd} shaded square.



The six statements in the body are given below, but not necessarily in the correct order.

```
1    if (n == 0) return;
2    drawShadedSquare(x, y, size);
3    draw(n-1, x - size/2, y + size/2, size/2.2);   // upper left
4    draw(n-1, x + size/2, y + size/2, size/2.2);   // upper right
5    draw(n-1, x - size/2, y - size/2, size/2.2);   // lower left
6    draw(n-1, x + size/2, y - size/2, size/2.2);   // lower right
```

The helper function `drawShadedSquare()` draws a gray square of side length `size` that is outlined in black and centered on `(x,y)`.

(a) Give a correct ordering of the statements above. Circle your answer.

(b) Circle those statement(s) below that are true for *every* correct ordering.

I. Statement 1 appears first.

II. Statement 2 appears before statements 3, 4, 5, and 6.

III. Statement 4 appears before statement 5.

IV. Swapping statements 3 and 6 produces another correct ordering.

V. Will result in a `StackOverflowError`.

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

Consider the following TOY code fragment.

```
20:  2AAB    R[A] <- R[A] - R[B]
21:  DA20    if (R[A]  > 0) pc <- 20
22:  CA24    if (R[A] == 0) pc <- 24
23:  1AAB    R[A] <- R[A] + R[B]
24:  0000    halt
```

(a) Suppose that just before the code fragment is executed, `R[A]` stores the value `001A` and `R[B]` stores the value `0008`. What are the values of `R[A]` and `R[B]` upon termination?

`R[A]:`

`R[B]:`

(b) Suppose that just before the code fragment is executed, `R[A]` stores the value `5EAB` and `R[B]` stores the value `0010`. What are the values of `R[A]` and `R[B]` upon termination?

`R[A]:`

`R[B]:`

(c) Give a one-line Java statement that corresponds to the TOY code fragment above, assuming that `R[A]` and `R[B]` contain positive integers and `a` and `b` are the corresponding Java `int` variables.

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.

16-bit registers (using two's complement arithmetic)
16-bit memory locations
 8-bit program counter
```