COS 126 Midterm 1 Written Exam Spring 2015

There are 9 questions on this exam, weighted as indicated below. The exam is closed book, though you are allowed to use a single-page one-sided hand-written cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. You will have 50 minutes to complete this test.

Print your name, login ID, and precept number on this page (now), and write out and sign the Honor Code pledge. Note, it is a violation of the Honor Code to discuss this midterm with **anyone** until after everyone in the class has taken the exam.

NAME: _____

LOGIN: _____

PRECEPT: _____

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

_____

_____

SIGNATURE: _____

#1	#2	#3	#4	#5	#6	#7	#8	#9	TOTAL
/8	/10	/4	/5	/6	/4	/10	/12	/11	**/70**

1. Number Representations (8 points). As many of you know, the number 42 is the Answer to the Ultimate Question of Life, the Universe, and Everything (at least according to "The Hitchhiker's Guide to the Galaxy").

Using 8-bit, 2's-complement notation, what is 42 (decimal) in binary?

00101010

What is 42 (decimal) in hexadecimal?

2A

Suppose 42 is a hex number. What is it in decimal?

66

Still using 8-bit, 2's-complement binary notation, how is -42 (decimal) written?

11010110

All the 42's from here on are decimal. **Express your answers in decimal**.

42 & 42 = 42

42 | 42 = 42

42 ^ 42 = 0

42 & -42 = 2

2. Arrays (10 points). Each of the following code snippets contains a print statement. However, some of them contain errors. For each of the snippets below, either categorize the error as a compilation error or runtime error, or, if there is no error, provide the output of the program.

A.

```
int[] a = new int[5];
System.out.println(a[5]);
```

- ☐ compilation error
- ☑ runtime error
- ☐ no error; the output is: _____

B.

```
int[] a = new int[4];
System.out.println(a.length);
```

- ☐ compilation error
- ☐ runtime error
- ☑ no error; the output is: 4

C.

```
int[] a = new int[4];
System.out.println(a[3]);
```

- ☐ compilation error
- ☐ runtime error
- ☑ no error; the output is: 0

D.

```
int[] a = {2, 4, 6, 8, 10};
int[] b = a;
System.out.println(b[3]);
```

- ☐ compilation error
- ☐ runtime error
- ☑ no error; the output is: 8

E.

```
int[] a; System.out.println(a[3]);
```

- ☑ compilation error
- ☐ runtime error
- ☐ no error; the output is: _____

F.

```
int[] a = {1, 2, 3, 4, 5};
a = new int[4];
for (int i = 0; i < a.length; i++)
    a[i] = 2 * i;
System.out.println(a[3]);
```

- ☐ compilation error
- ☐ runtime error
- ☑ no error; the output is: 6

3. Methods (4 points). Consider the following code.

```
public class MethodTester {
    private static void methodB(int[] c, int d) {
        c[0]++;
        d += 42;
    }

    private static int methodA(int[] a, int b) {
        methodB(a, b);
        a[0]++;
        return b/2;
    }

    public static void main(String[] args) {
        int[] arr = {8, 9, 10};
        int x = 1;
        x = methodA(arr, x);
        System.out.println(arr[0] + " " + x);
    }
}
```

Which one of the following is the output of this program? Circle your answer.

"8 3"

"8 10"

"8 21"

"9 1"

"9 3"

"9 21"

"10 0"

"10 1"

"10 21"

4. Conditionals & Loops (5 points). Match each of the following code snippets with a one-line statement that would accomplish the same thing. Fill in the blanks after each code snippet with one letter from the following list:

A. `int c = 0;`
B. `int c = 1;`
C. `int c = N/2;`
D. `int c = N;`
E. `int c = N + 1;`
F. `int c = N + N;`
G. `int c = N * N;`
H. `int c = (int)(Math.random() * 60);`
I. `int c = 10 + ((int) (Math.random() * 60));`
J. `int c = 10 * ((int) (Math.random() * 6));`
K. `int c = 10 * (1 + (int) (Math.random() * 6));`
L. `int c = (int) (Math.random()) * 60;`
M. `int c = ((int) (Math.random()) + 1) * 60;`

```
int c = 0;
for (int i = 0; i < N; i++)
    c++;
```

Accomplishes the same thing as: D

```
int c = 0;
while (N > 1) {
    N = N - 2;
    c++;
}
```

Accomplishes the same thing as: C

```
double x = Math.sqrt(N + N);
x = Math.abs(x);
int c = (int) (x * x);
```

Accomplishes the same thing as: F

```
int c;
c = (int)(Math.random() * 6);
if (c < 1) c = 10;
else if (c < 2) c = 20;
else if (c < 3) c = 30;
else if (c < 4) c = 40;
else if (c < 5) c = 50;
else            c = 60;
```

Accomplishes the same thing as: K

```
int c = 0;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        c++;
    }
}
```

Accomplishes the same thing as: G

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.
pc starts at 10

16-bit registers
16-bit memory locations
8-bit program counter

5. TOY (6 points). The TOY program below reads two integers from TOY's StdIn. It computes their integer quotient by repeatedly subtracting the second number from the first, counting the number of subtractions that are performed. It prints the number of subtractions to TOY's StdOut. The remainder, if any, is ignored.

Fill in the blanks to complete the program and the pseudocode.

```
10: 8AFF   read R[A]
11: 8BFF   read R[B]
12: 7C00   R[C] <- 0000
13: 7101   R[1] <- 0001
14: 2AAB   R[A] <- R[A] - R[B]
15: 1CC1   R[C] <- R[C] + R[1]
16: DA14   if (R[A] > 0) goto 14
17: CA19   if (R[A] == 0) goto 19
18: 2CC1   R[C] <- R[C] - R[1]
19: 9CFF   write R[C]
1A: 0000   halt
```

6. Debugging (4 points). Here is a buggy Java method that is supposed to convert a positive integer into its binary equivalent, in String form.

```
1  public class BuggyConversion {
2      public static String convertNR (int x) {
3          String s = "1";
4          while (x > 1) {
5              s = (x % 2) + s;
6              x = x/2;
7          }
8          return s;
9      }
10 }
```

Sometimes it gets the right answer. For instance, 7 is converted to "111". But, mostly, it gets the wrong answer: 6 is converted to "101" and 5 is converted to "011". When you began debugging this program you found that one problem was the initialization of String s. So you wisely decided that s should be initialized to the empty String. So you replace line 3 with:

```
String s = "";
```

You compile, and start testing. But now, 7 is converted to "11" and 6 is converted to "10" and 5 is converted to "01".

One simple bug remains. What is it? Identify the line on which it occurs, and write the replacement line of code.

LINE NUMBER:

4

REPLACEMENT LINE OF CODE:

while (x > 0) {

Alternative solution: replace line 8 with `return 1 + s;`

7. Sorting (10 points). Two sorting algorithms, Insertion Sort and Merge Sort, will be used to sort the following array of characters into alphabetical order, left-to-right:

M E L T S N O W

Assume that Insertion Sort will scan the array from left to right, and that Merge Sort will recursively sort the left side before the right.

The following array contents may occur at some point during either, both, or none of these sorts. **Check all that apply.**

	Occurs During Insertion Sort	**Occurs During Merge Sort**	**Does <u>Not</u> Occur During Either**
E M L T S N O W	☑	☑	☐
S M E L T N O W	☐	☐	☑
E L M T N O S W	☐	☑	☐
E L M S T N O W	☑	☐	☐
E L M T S N O W	☑	☑	☐

8. Recursion Meets I/O (12 points). Consider the following code.

```
public class IOFun {
    public static int merge(String[] args, int i) {
        int x = Integer.parseInt(args[i]);
        if (i == 1) return x;
        return merge(args, i-1) + x;
    }

    public static void main(String[] args) {
       int n = Integer.parseInt(args[0]);
       String s = "";
       for (int i = n; i > 0; i--)
           s = s + StdIn.readString() + ",";
       int x = merge(args, args.length-1);
       StdOut.println(x + "@" + s);
    }
}
```

Suppose the file **input.txt** contains:

```
mind
like
parachute
only functions when open
```

What is printed when the user runs the following commands?

% java-introcs IOFun 2 3 < input.txt

3@mind,like,

% java-introcs IOFun 4 29 7 6 < input.txt

42@mind,like,parachute,only,

9. Recursive Graphics (11 points). Complete the following recursive graphics program using the following lines of code such that the program produces each of the pictures below. For the sake of simplicity, you do not have to worry about setting the pen color.

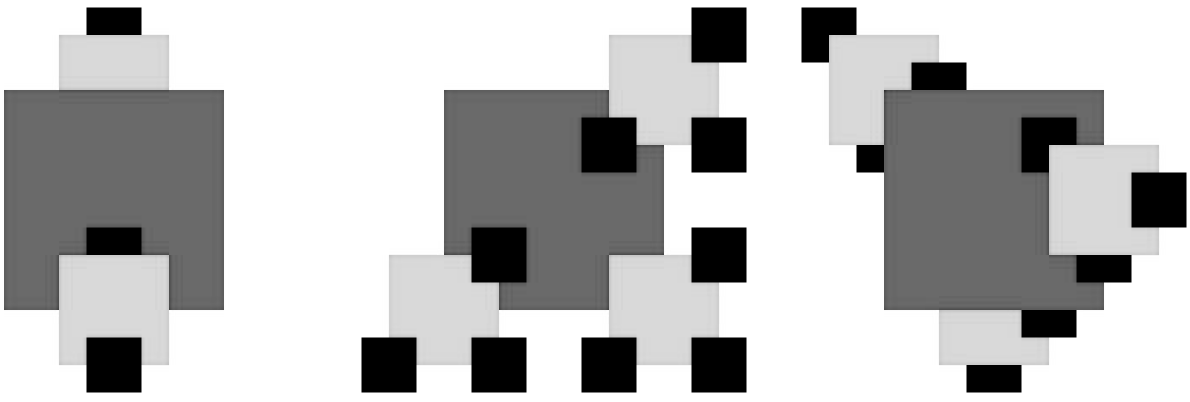
```
public class Squares {
    public static void recur(int N, double x, double y, double s) {
        if (N == 0) return;

        // What goes here?
    }

    public static void main(String[] args) {
        recur(3, .5, .5, .25);
    }
}
```

A. `recur(N-1, x+s, y,   s/2);     // Recur right`
B. `recur(N-1, x-s, y,   s/2);     // Recur left`
C. `recur(N-1, x,   y+s, s/2);     // Recur top`
D. `recur(N-1, x,   y-s, s/2);     // Recur bottom`
E. `recur(N-1, x+s, y+s, s/2);     // Recur top right`
F. `recur(N-1, x-s, y+s, s/2);     // Recur top left`
G. `recur(N-1, x+s, y-s, s/2);     // Recur bottom right`
H. `recur(N-1, x-s, y-s, s/2);     // Recur bottom left`
I. `StdDraw.filledSquare(x, y, s); // Draw square at (x, y)`

In the boxes below, provide **an ordered sequence of 3-4 letters** corresponding to the 3-4 lines of code that you would need to add to the recur() method to produce each of these pictures.



C	I	D

I	E	G	H

(E, G, H) in any order

F	D	I	A

(F, D) in either order

This page is provided as scratch paper. Tear it out and return it inside the exam when finished.

NAME: _____

LOGIN: _____

PRECEPT: _____