COS 126 Written Exam 1 Fall 2019

There are eight questions on this exam. There is one question per lecture, numbered corresponding to the lectures, *not in order of difficulty*. If a question seems difficult to you, skip it and come back to it. You will have 50 minutes to complete the exam.

This exam is preprocessed by computer. If you use pencil (and eraser), write darkly. Write all answers inside the designated rectangles. When asked to fill in circles, do so completely with a dark pencil. If you change your mind, you must erase completely and fill in another circle!

Do this ● not this ✓ or this ✗ or this ✗.

Resources. You may reference your optional one-sided 8.5-by-11 handwritten "cheat sheet" during this exam. You may not use the textbook, your notes, or any electronic devices. You may not communicate with anyone except the course staff during this exam.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam later. Do not discuss its contents with anyone who has not taken it.

This page. Do not remove this exam from the exam room. Fill in this page now, but do not start the exam until you are told.

Name	
NetID	
Precept	
Exam Room	

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

_____

_____

[*copy the pledge here*]

_____

[*signature*]

Q1a. Types and Casts (1 point). In each row, fill in the circle corresponding to the value of the given expression. You must fill in exactly one circle in each row.

	`0`	`1`	`0.0`	`1.0`	`NaN`	*won't compile*
`3 - (int) "2.0"`	○	○	○	○	○	○
`4 / 6 * 1.5`	○	○	○	○	○	○
`1 / 1-1 / 1`	○	○	○	○	○	○
`1.5 * 4 / 6`	○	○	○	○	○	○

Q1b. Terminology (1.5 points). Put the letter to left of each definition that identifies the term defined. No letter may be used more than once.

☐ Rules that determine in which order to apply operations

☐ Step-by-step description of the operation of a program

☐ A finite sequence of characters

☐ Data structure that holds a sequence of values of the same type

☐ Source-code representation of a data-type value

☐ A set of values and operations on those values

A array

B data type

C literal

D precedence

E string

F trace

G variable

H *none of the above*

Q2. Loops and conditionals (2 points). Suppose that this code is in the file `Q2.java`

```
public class Q2
{
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        for (int j = -N; j <= N; j++)
        {
            for (int i = -N; i <= N; i++)
            {
                if      (i == j)  System.out.print("A");
                else if (i == -j) System.out.print("B");
                else              System.out.print("C");
            }
        }
        System.out.println();
    }
}
```

and that you compile it using the command `javac-introcs Q2`. Fill in the circle corresponding to the specified character. You must fill in exactly one circle in each row.

	A	B	C
first character printed by `java-introcs Q2 0`	○	○	○
second character printed by `java-introcs Q2 1`	○	○	○
ninth character printed by `java-introcs Q2 2`	○	○	○
twenty-fifth character printed by `java-introcs Q2 3`	○	○	○

Q3a. Arrays (1 point). Consider the following Java program

```
public class Q3a
{
    public static void main(String[] args)
    {
        int[] a = new int[6];
        int[] b = new int[a.length];
        b = a;
        for (int i = 1; i < b.length; i++)
            b[i] = i;
        for (int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println();
    }
}
```

Fill in the *one* circle at right that corresponds to the output of this program

`0 0 0 0 0 0` ○

`0 1 2 3 4 5` ○

cannot be determined from the information given ○

Q3b. Two-dimensional arrays (1 point). Put the letter to left of each definition that describes the given code. No letter may be used more than once.

☐ Refers to the number of rows

☐ Refers to a specified row

☐ Declares a two-dimensional array

☐ Creates a two-dimensional array

A `double[][] a;`

B `a.length`

C `a.row(i)`

D `a[i][j]`

E `a = new double[2][2];`

F `a[i]`

H *none of the above*

Q4. I/O (2 points). *Carefully study* the following Java program. It is simple, but there is a likely bug that makes it tricky.

```
public class Q4
{
    public static void main(String[] args)
    {
        int x = StdIn.readInt();
        int y = Integer.parseInt(args[0]);
        StdOut.print(x + y + " ");
        StdOut.print(x + " ");
        StdOut.println(y);
    }
}
```

Suppose that the file `input.txt` contains the string `"6 5 4 3"`. In the box to the left of each command, write the letter corresponding to the output it produces. A letter may be used once, more than once, or not at all.

☐ `java-introcs Q4 3`

☐ `java-introcs Q4 3 < input.txt`

☐ `java-introcs Q4 3 > input.txt`

☐ `java-introcs Q4 3 < input.txt | java-introcs Q4 2`

A `6 2 6`

B `11 9 2`

C `6 5 4 3`

D `9 6 3`

E `6 3 6 3`

F *none of the above*

Q5. Functions (2 points). Consider the following code.

```
public class Cubes
{
    public static int square(int i)
    { return i * i * i; }

    public static void main(String[] args)
    {
        for (int i = 1; i <= 1000; i++)
            StdOut.println(square(i));
    }
}
```

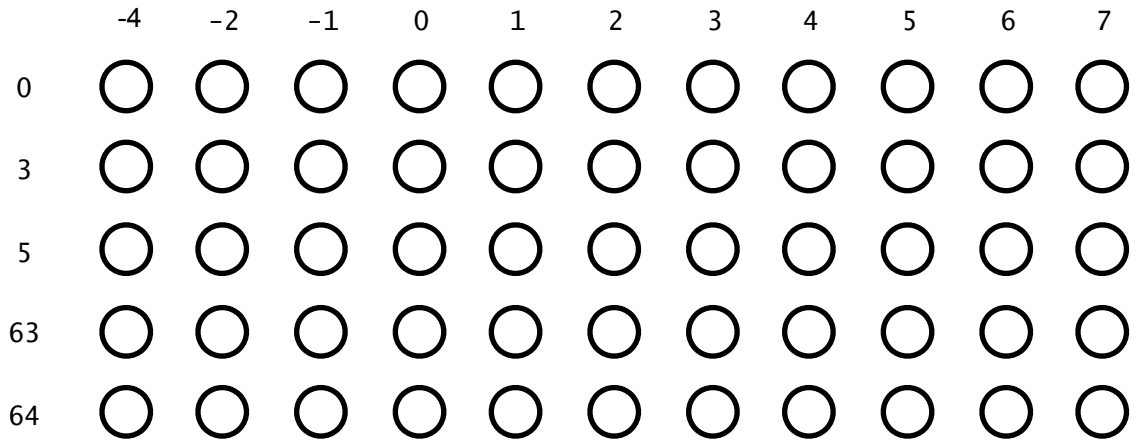
Fill in the circle to the right of each statement to indicate whether it is true or false.

	true	*false*
Will not compile because braces in `square()` are not on separate lines.	○	○
Will not compile because braces are missing in the `for` loop.	○	○
Will not compile because `i` is not declared in `square()`.	○	○
Prints the squares of the integers from 1 to 1000.	○	○
Prints the cubes of the integers from 1 to 1000.	○	○
Goes into an infinite loop.	○	○
Prints only a few lines because `square()` rapidly increases the `i` used by `main()`.	○	○
Causes a runtime error because of a type mismatch	○	○

Q6. Recursion (2.5 points). Suppose that this code is in the file `Q6.java`

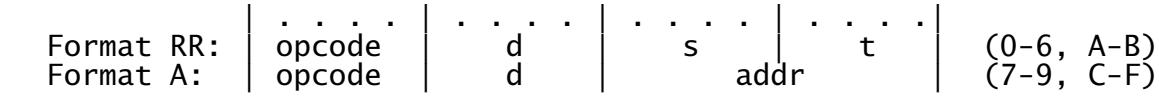
```
public class Q6
{
    public static int f(int x)
    {
        if (x < 2) return x;
        if (x % 2 == 1) // x is odd
            return f((x - 1) / 2) - 1;
        else            // x is even
            return f(x / 2) + 1;
    }

    public static void main(String[] args)
    {
        int n = Integer.parseInt(args[0]);
        StdOut.println(f(n));
    }
}
```

Now suppose that you compile this program using the command `javac-introcs Q6` and then run it using the command `java-introcs Q6 x` for the various values of `x` given at left below. Fill in the circle corresponding to the value printed for each given value of `x`. You must fill in exactly one circle in each row.



TOY REFERENCE CARD

INSTRUCTION FORMATS



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] <- M[addr]
9:	store	M[addr] <- R[d]
A:	load indirect	R[d] <- M[R[t]]
B:	store indirect	M[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 M[FF] come from stdin.
Stores to M[FF] go to stdout.

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

Q7. TOY (2 points). In the box to the left of each description, write the letter corresponding to the four-digit hex number that best fits the description. A letter may be used once, more than once, or not at all.

☐ Represents the decimal integer 20

☐ Represents the decimal integer -20

☐ Halt instruction

☐ Doubles the value in a register

☐ Writes to standard output

☐ Illegal TOY instruction

☐ Represents the decimal integer 4369

☐ No-op (has no effect)

A	0014
B	0020
C	1020
D	1110
E	1111
F	1212
G	4369
H	9FFF
I	D099
J	FFEC
K	*none of the above*

Q8. TOY Programming (2.5 points). An integer array is stored starting at memory location 51. The length of the array is stored in memory location 50. The following (partial) TOY program reverses the array by swapping the first element and the last element, then swapping the second element and the next-to-last element, and so on until the `lo` and `hi` pointers cross.

```
10: 7101   R[1] <- 01               R[1] gets constant 1
11: ____   R[2] <- 51               addr of first element (lo)
12: ____   R[3] <- M[50]
13: 1323   R[3] <- R[2] + R[3]
14: 2331   R[3] <- R[3] - R[1]      addr of last element (hi)
15: ____
16: D41E   if (R[4] > 0) pc <- 1E   15-16 exit loop if hi < lo
17: ____   R[5] <- M[R[2]]
18: A603   R[6] <- M[R[3]]
19: B503   M[R[3]] <- R[5]
1A: B602   M[R[2]] <- R[6]
1B: 1221   R[2] <- R[2] + R[1]      increment lo pointer
1C: 2331   R[3] <- R[3] - R[1]      decrement hi pointer
1D: ____
1E: 0000   halt
```

For each of the missing instructions, write the letter corresponding to the correct instruction in the box to the left of the given address.

11: ☐

12: ☐

15: ☐

17: ☐

1D: ☐

A	2423
B	2432
C	7251
D	7350
E	8250
F	8251
G	8350
H	A502
I	B502
J	C015