COS126 Written Exam 2 (Spring 2015)

There are 8 questions on this exam, weighted as indicated below. This exam is closed book. You may use a single-page two-sided hand-written cheatsheet. There is a blank page intended for scratch paper at the end of the exam. No calculators or other electronic devices are allowed. Give your answers and show your work in the space provided. You will have 50 minutes to complete this test.

Print your name, login, and precept information on this page (now). Write out and sign the Honor Code pledge. As you know, it is a violation of the Honor Code to discuss the contents of this exam **with anyone** until after the solutions have been posted.

NAME: _____

LOGIN: _____

PRECEPT: _____

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

_____

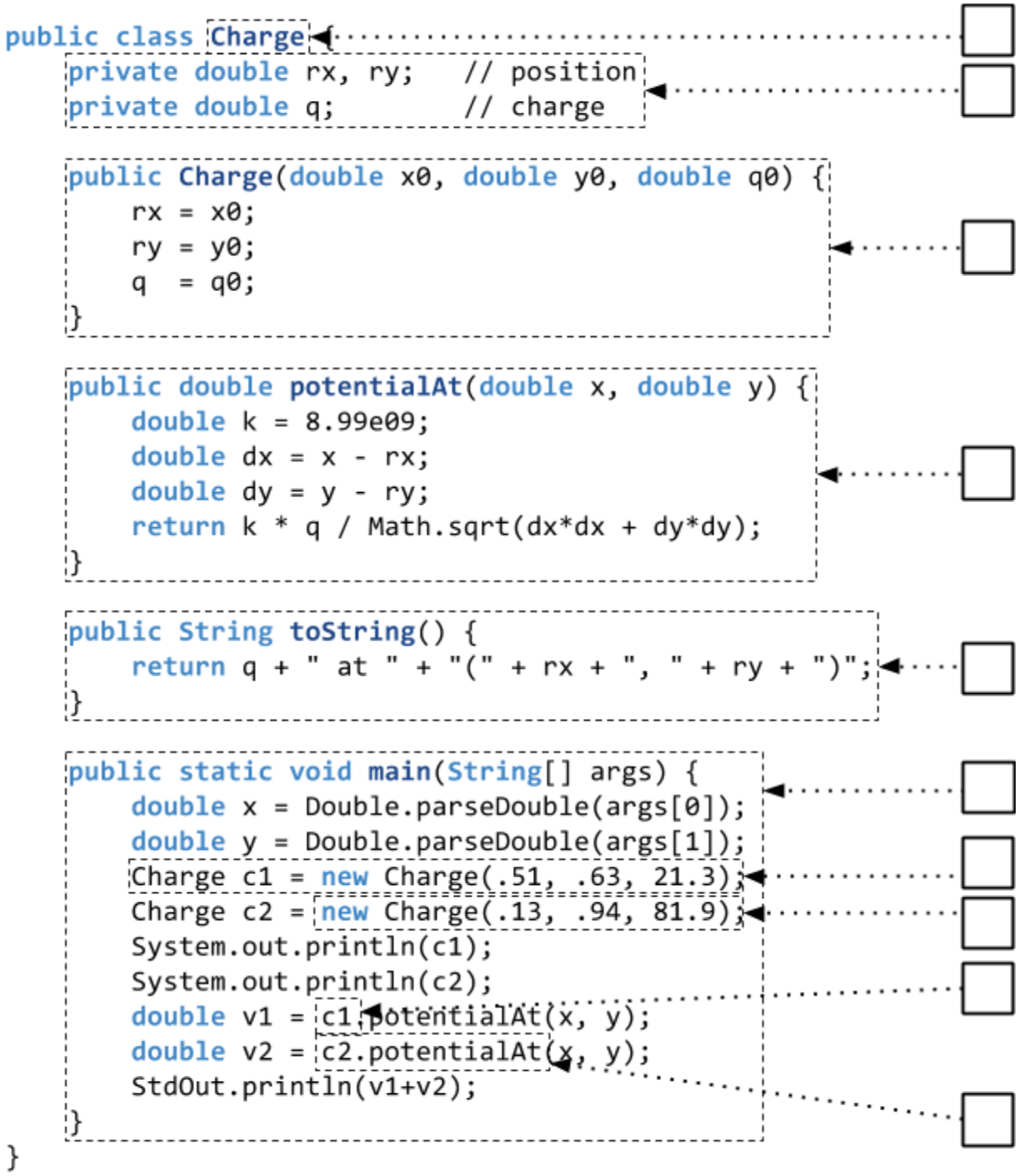
_____

SIGNATURE: _____

#1	#2	#3	#4	#5	#6	#7	#8	TOTAL
/10	/5	/10	/8	/14	/10	/8	/5	**/70**

1. Object Oriented Programming (10 points). Fill in the boxes below with the letter corresponding to the best description of that part of the program.

A. initialize object	**D.** invoke method	**G.** instance variables
B. local variable	**E.** constructor	**H.** class name
C. instance method	**F.** invoke constructor	**I.** test client



2. Abstract Data Types (5 points). Choose the best data type for the following situations.

A. `ST<String[], String>`

B. `ST<String, Queue<String>>`

C. `ST<Integer, Integer>`

D. `Queue<String>`

E. `ST<String, ST<String, int[]>`

F. `ST<Integer, Point>`

G. `Stack<Point>`

H. `Queue<Point>`

1. For each precept of COS 126 (named P01, P02, P02A, etc.), store the grades that each student (netID) gets for Assignments 0 through 8. ☐

2. Keep track of all your U.S. friends by state. ☐

3. For each day of a given month, record the number of COS 126 students who have an exam on that day. ☐

4. Make a list of all the books you plan to read this summer. (Assume that there are at least 2 books.) ☐

5. In a graph of Points, after finding the shortest path from some Point `start` to another Point `finish`, find the Point-by-Point directions for the path from `finish` back to `start`. ☐

3. Regular Expressions (10 points). Choose the regular expression that matches the following languages. The alphabet of all of these languages is `{0, 1}`.

A. `1*001*`

B. `(01)*|(10)*`

C. `(01)+|(10)+`

D. `1*01+01*`

E. `(0|1)*000`

F. `(0|1)(0|1)*(0|1)`

G. `0|1|(0(0|1)*0)|(1(0|1)*1)`

H. `0|00|((0|1)*000)`

I. `(0|1)*01+0(0|1)*`

1. All strings beginning and ending with the same character (including the strings "`0`" and "`1`"). ☐

2. All strings containing at least two nonconsecutive `0`'s. ☐

3. All strings containing exactly two nonconsecutive `0`'s and no other `0`'s. ☐

4. All strings that correspond to binary numbers divisible by 8. ☐

5. All even length strings with only alternating `1`'s and `0`'s, including the empty string. ☐

4. Linked Lists (8 points). Consider the following code for a null-terminated linked list.

```
public class GuitarList {
    private Node start; // first node in the linked list

    private class Node {
        private GuitarString value;
        private Node next;
    }

    public GuitarList() { start = null; }

    // puts n at the end of the linked list
    public void insertAfterLast(Node n) {
        if (start == null) { // special case
            start = n; return;
        }
        // finish this method
    }
}
```

Which of the following would be a successful implementation of `insertAfterLast()`?

```
Node i = start;
do {
    i = i .next;
} while (i != null);
i.next = n;
```

☐ **YES** ☐ **NO**

```
for (Node i = start; i.next != null; i = i.next);
i.next = n;
```

☐ **YES** ☐ **NO**

```
Node i = start;
Node j = start;
while (i != null) {
    j = i;
    i = i.next;
}
j.next = n;
```

☐ **YES** ☐ **NO**

```
Node i = start;
boolean keepGoing = true;
while (keepGoing) {
    Node next = i.next;
    if (next == null)
        keepGoing = false;
    else
        i = next;
}
i.next = n;
```

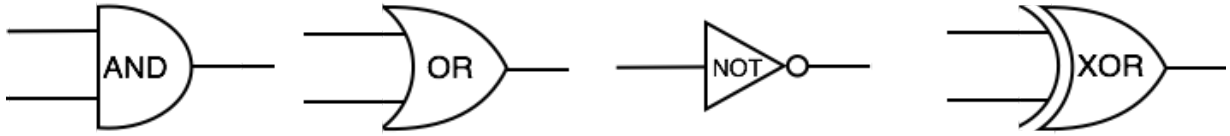
☐ **YES** ☐ **NO**

5. Theory (14 points). For each statement below, check all that apply.

	The Halting Problem	Factoring	Sorting	Boolean Satisfiability (SAT)	Travelling Salesperson Problem
1. These problems are decidable/computable.	☐	☐	☐	☐	☐
2. As far as we know, only these problems are in P.	☐	☐	☐	☐	☐
3. As far as we know, these problems are in NP but not in P.	☐	☐	☐	☐	☐
4. These problems can be reduced to SAT.	☐	☐	☐	☐	☐
5. As far as we know, these problems are not in P and not NPC.	☐	☐	☐	☐	☐

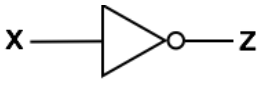
	Turing Machines	Regular Expressions	TOY with sufficient memory	DFA's	None of these
6. If the Church-Turing thesis holds, these models can perform any possible computation.	☐	☐	☐	☐	☐
7. These models cannot express some Java programs.	☐	☐	☐	☐	☐
8. These models cannot be simulated in Java.	☐	☐	☐	☐	☐
9. These models always halt on all finite inputs.	☐	☐	☐	☐	☐
10. These models can always correctly check whether an arbitrary Java program halts on some finite input.	☐	☐	☐	☐	☐

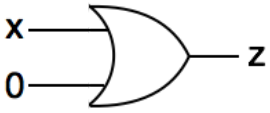
6. Circuits (10 points). First, a reminder of the shape of the following logic gates:

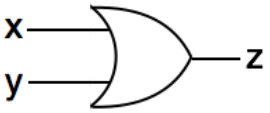


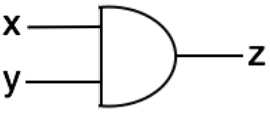
In the left column below, there are circuits that use only **AND**, **OR**, and **NOT** gates.
In the right column below, there are circuits that use only **XOR** gates.

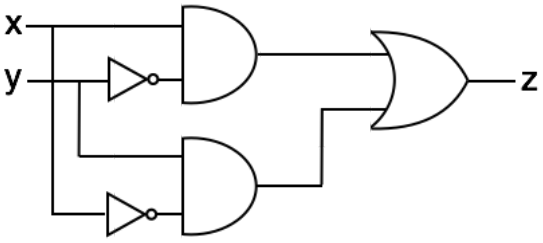
Fill in the boxes on the left with the letter of the equivalent circuit on the right.
If no circuit on the right matches, put the letter "F" in the box.

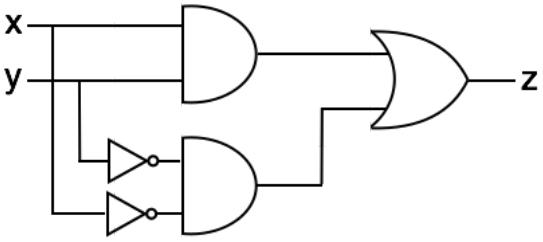
☐ 

☐ 

☐ 

☐ 

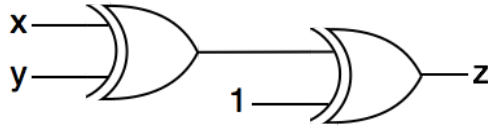
☐ AND ☐ 

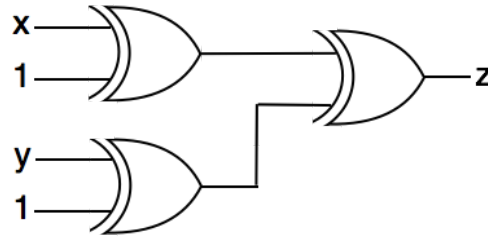
☐ 

A 

B 

C 

D 

E 

F **NONE OF THE ABOVE**

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]
```

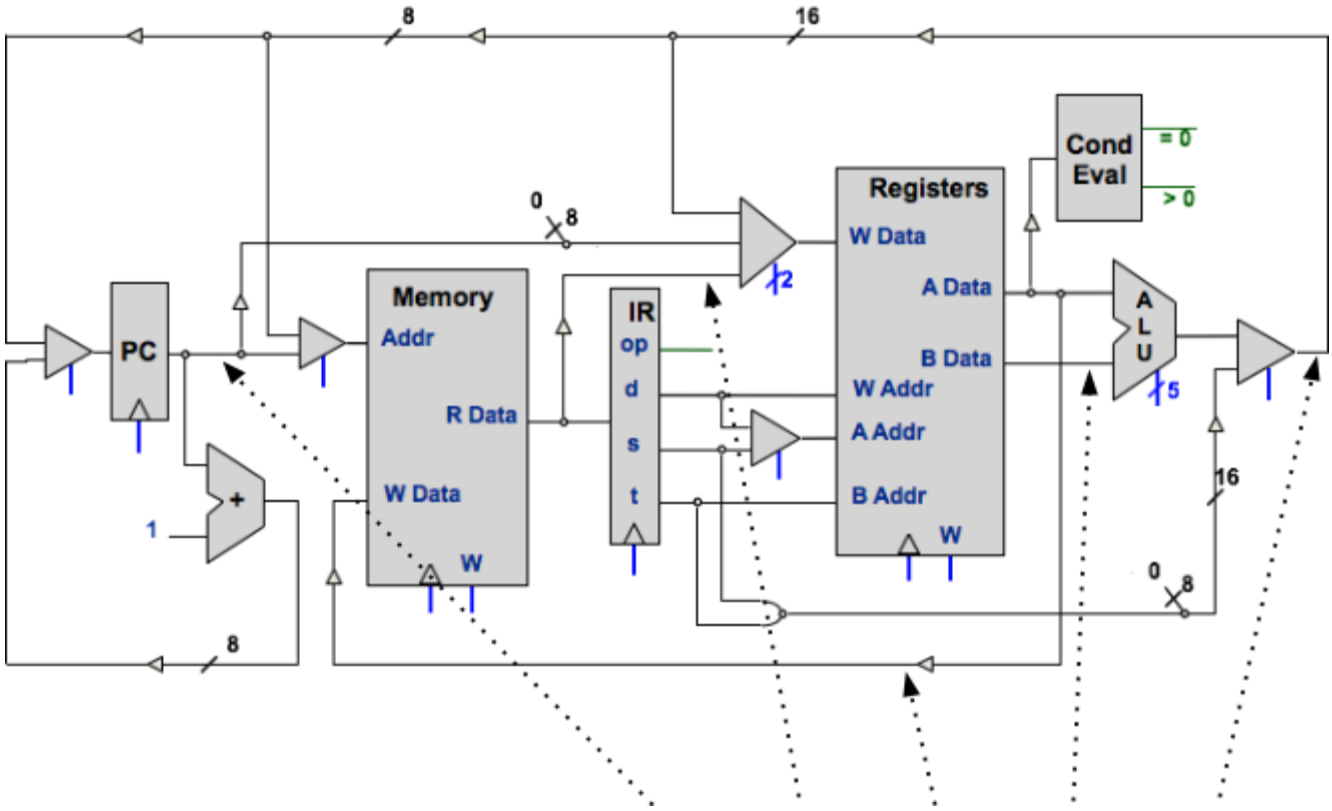
CONTROL

```
    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

7. TOY architecture (8 points). Check all the boxes that correspond to the datapaths needed to fetch and execute the following TOY instructions.



	A	B	C	D	E
AND `R[d] <- R[s] & R[t]`	☐	☐	☐	☐	☐
LOAD `R[d] <- mem[addr]`	☐	☐	☐	☐	☐
LOAD ADDRESS `R[d] <- addr`	☐	☐	☐	☐	☐
STORE `mem[addr] <- R[d]`	☐	☐	☐	☐	☐

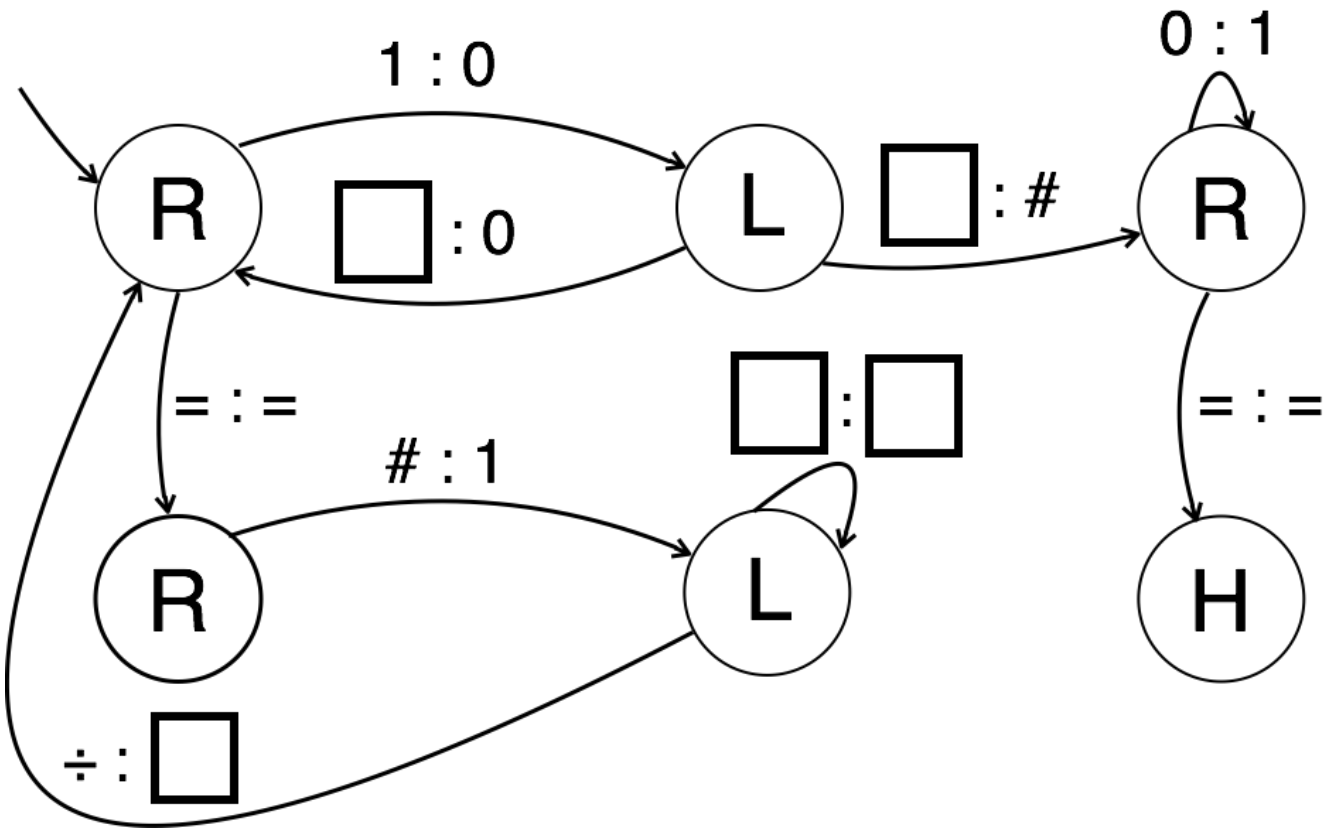
8. Turing Machines (5 points). The Turing Machine below will perform integer division on unary numbers, discarding the remainder. As you will recall, unary numbers are represented by consecutive ones. For example 4 ÷ 2 = 2, in unary, is expressed as follows: 1111 ÷ 11 = 11. Your task is to fill in the blank squares below so that this machine will work. As usual, we omit arrows that stay in the same state and don't change the symbol.

For example, if the tape is configured as follows, with the tape head on the ÷ symbol.

...	#	#	1	1	1	1	1	1	1	÷	1	1	1	=	#	#	#	#	...

The final state of the tape should be:

...	#	#	1	1	1	1	1	1	1	÷	1	1	1	=	1	1	#	#	...



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

NAME: _____

LOGIN: _____

PRECEPT: _____