COS126 **Written Exam 2** **Spring** '19

Instructions. This exam has 7 questions, worth 10 points each. You will have 50 minutes.

Resources. You may use your optional two-sided 8.5-by-11 handwritten reference 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.

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

This paper. Do not remove this copy of the exam from the exam room. You may fill in this page now.

NAME: _____

NETID: _____

PRECEPT: _____

EXAM ROOM: _____

"I pledge my honor that I will not violate the Honor Code during this examination."

_____

_____

SIGNATURE: _____

Question 1	**Number Systems**	**10** points

Perform the following base conversions.

1. Convert the decimal value 100 to binary.

---

2. Convert the binary value 100 to decimal.

---

3. Convert the hexadecimal value 100 to binary.

---

4. Convert the decimal value 100 to hexadecimal.

---

5. Convert the binary value 100 to hexadecimal.

Perform the following bitwise operations. All values are in hexadecimal. Express your answers in hex.

6. `0001 & 0010`

---

7. `1111 | 0000`

---

8. `C0DE ^ 100`

---

9. `(1001 ^ 1110) | 1100`

---

10. `(1001 ^ 1110) | (1100 & (1110 ^ 1001))`

Question 2	**Regular Expressions**	**10** points

Let `L = {01110, 11100, 10111, 01111}`. The alphabet for `L` is `{0, 1}`.

For each of the following regular expressions, choose one of the following:

NONE	Matches no strings in `L`.
LESS	Matches at least one string in `L`, but not all, and no strings that are not in `L`.
POOR	Matches at least one string in `L`, but not all, and at least one other string.
EXACT	Matches all strings in `L` and no strings that are not in `L`.
MORE	Matches all strings in `L` and at least one other string.

		NONE	LESS	POOR	EXACT	MORE
1.	`(10)*(01)*`	○	○	○	○	○
2.	`(1|0)*`	○	○	○	○	○
3.	`(0|1)11.*`	○	○	○	○	○
4.	`01110|01111`	○	○	○	○	○
5.	`..1..`	○	○	○	○	○
6.	`(0111.)|(1.1(00|11))`	○	○	○	○	○
7.	`1*0*1*0*`	○	○	○	○	○
8.	`((10|01)111)|(0111(0|1))|11100`	○	○	○	○	○
9.	`(0|1)1(0|1)1(0|1)1(0|1)`	○	○	○	○	○
10.	`(0|1)*1(0|1)*1(0|1)*1(0|1)`	○	○	○	○	○

Question 3	**TOY**	**10** points

For each TOY program below, indicate which operation (**A-J**) is performed on the contents of `R[A]`. Assume each TOY program is independent of one another. Ignore integer overflow. Select the best answer for each program. You may use each operation (**A-J**) once, more than once, or not at all.

A. No-op (no change).	**F.** Add 1.
B. Flip all bits (0s to 1s and 1s to 0s).	**G.** Subtract 1.
C. Multiply by 2.	**H.** Shift left 2.
D. Divide by 2.	**I.** XOR with all 1s.
E. AND with 1.	**J.** Negate (multiply by -1).

1.
```
1AAA  R[A] <- R[A] + R[A]
```
☐

2.
```
4AAF  R[A] <- R[A] ^ R[F]
4AAF  R[A] <- R[A] ^ R[F]
```
☐

3.
```
7101  R[1] <- 0001
5AA1  R[A] <- R[A] << R[1]
```
☐

4.
```
7101  R[1] <- 0001
2101  R[1] <- R[0] - R[1]
3AA1  R[A] <- R[A] & R[1]
```
☐

5.
```
7101  R[1] <- 0001
2B01  R[B] <- R[0] - R[1]
4AAB  R[A] <- R[A] ^ R[B]
1AA1  R[A] <- R[A] + R[1]
```
☐

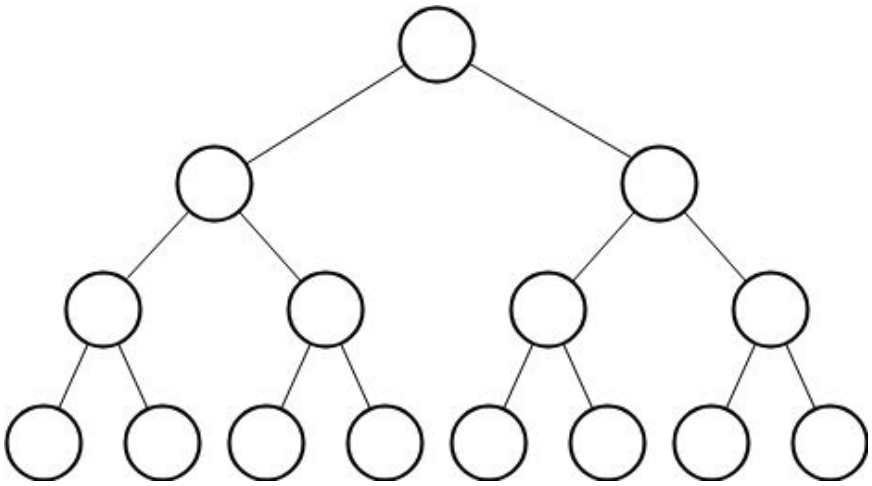
Question 4	**Theory**	**10** points

Assess whether the following statements are true, false, or currently unknown.

		true	false	unknown
1.	TSP can be solved efficiently.	○	○	○
2.	If you can write a program to compute something in Java, you can create a Turing Machine to perform the same computation.	○	○	○
3.	If you can create a Turing Machine to compute something, you can write a Java program to perform the same computation.	○	○	○
4.	According to the extended Church-Turing thesis, every problem that can be solved efficiently by a Turing Machine can be solved efficiently by any other model of computation.	○	○	○
5.	It is possible to decide whether some programs halt.	○	○	○
6.	A DFA can get stuck in an infinite loop.	○	○	○
7.	If a polynomial time solution is found to FACTOR, then P = NP.	○	○	○
8.	If an efficient solution is found to FACTOR, there must be a polynomial-time reduction from SAT to FACTOR.	○	○	○
9.	Every problem that is NP-Complete polynomial-time reduces to every other problem that is NP-Complete.	○	○	○
10.	Any problem polynomial-time reduces to itself.	○	○	○

Question 5	**BSTs & STs**	**10** points

Fill in the nodes below by inserting these values in a BST, in this order: 81, 92, 28, 28, 70, 40, 17, 8, 19. For full credit, you must leave any unused nodes blank. (If you make a mistake, just cross it out.)



Fill in the blanks in the following Java program that stores words and their definitions (one per word).

```
public class Dictionary {
   _________________ ST<String, String> dictionary = new _______________________________;

   public void addWord(String word, String definition) {
      dictionary.put(word, definition);
   }

   // returns the definition of "word"
   public String getDefinition(String word) {
      if (_______________________________) throw new RuntimeException("word not found");
      return _______________________________;
   }

   // returns all words and definitions, in the format: "word1: definition1\nword2: definition2\n"...
   public String toString() {
      StringBuilder sb = new StringBuilder();
      for (_________________ word : _______________________________) {
         sb.append(word + ": ");
         sb.append(_______________________________ + "\n"); // definition
      }
      return _______________________________;
   }
}
```

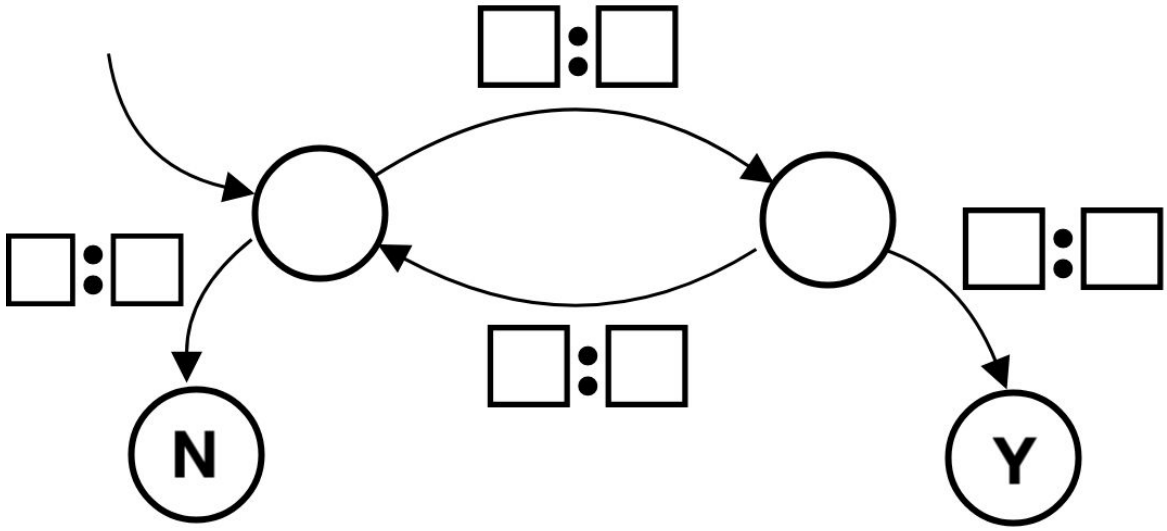
Question 6	**Turing Machines**	**10** points

Fill in the blanks in the following Turing Machine that accepts strings with an odd number of 1s.

Assume the following:

- The tape contains a binary string (i.e., one group of consecutive 1s and 0s, in any order).
- As usual, on either side of the input, there are an infinite number of hashes (#).
- The tape-head starts on the left-most bit.

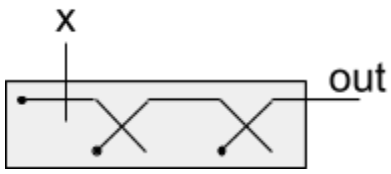
For full credit, your Turing Machine should not change the contents of the tape.



Don't forget to fill in both blank states above!

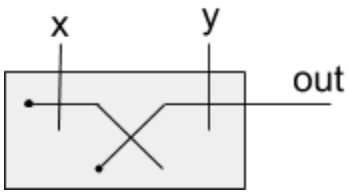
Question 7	**Circuits**	**10** points

For each circuit below, complete the corresponding truth table and choose the best descriptor.



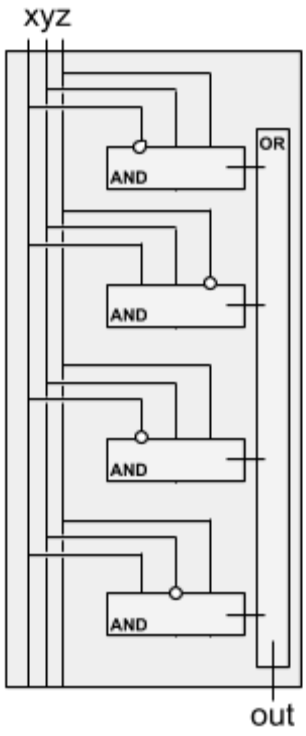
x	out
0	
1	

◯ not ◯ maj
◯ xor ◯ odd
◯ and ◯ even
◯ none of the above



x	y	out
0	0	
0	1	
1	0	
1	1	

◯ not ◯ maj
◯ xor ◯ odd
◯ and ◯ even
◯ none of the above



x	y	z	out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

◯ not ◯ maj
◯ xor ◯ odd
◯ and ◯ even
◯ none of the above

TOY REFERENCE CARD

INSTRUCTION FORMATS

```
             | . . . . | . . . . | . . . . | . . . .|
  Format RR: | opcode  |    d    |    s    |    t   |  (0-6, A-B)
  Format A:  | 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] <- 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 (two's complement)

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

You may tear this page out and use it as scratch paper.

If you do, please do *not* return this page. (Please recycle it after the exam.)