Instructions. This exam has eight (8) questions worth a total of one hundred (100) points. You have fifty (50) minutes.

This exam is preprocessed by computer. Write neatly and legibly. If you use a pencil, write darkly. Write all answers inside the designated rectangles and nothing else (e.g., no scratch work inside designated rectangles). Fill in circles completely: ⚫ (not ✔ or ✘). If you change your mind, you must erase completely and fill in another circle!

Resources. The exam is closed book, except that you are allowed to use a single two-sided reference sheet (8.5-by-11 paper, two-sided, in your own handwriting). No electronic devices are permitted.

Discussing this exam. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. Do not remove this exam paper from this room. Print your name, NetID, precept, and the room in which you are taking the exam in the space below. Also, **copy and sign the Honor Code pledge**. **(1 point).** You may enter this information now. Again, please write neatly and legibly.

NAME:

NETID (not email alias):

PRECEPT:

EXAM ROOM:

○ McCosh 50 ○ McDonnell A02 ○ CS 104

○ CS 301/302/401 ○ OTHER ___________________

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

SIGNATURE:

Fill in **ALL** the statement numbers where the described entities appear. An example is provided.

```
1 public class Ball {
2
3   private double rx, ry;
4
5   public Ball() {
6     rx = 0.5; ry = rx;
7   }
8
9   public void move() {
10    rx = rx + .001;
11    ry = ry + .002;
12  }
13
14  public void move(double x, double y) {
15    rx = rx + x;
16    ry = ry + y;
17  }
18  public Ball copy() {
19    Ball b = new Ball();
20    b.rx = rx;
21    b.ry = ry;
22    return b;
23  }
24
25  public static void main(String[] args) {
26    Ball b1 = new Ball();
27    Ball b2 = new Ball();
28    Ball b3 = b2;
29    Ball b4 = b3.copy();
30    b1.move();
31    b2.move(.1, .1);
32    b3.move();
33    b4.move();
34  }
35 }
```

*Example: the **+** operator (appears in statements 10, 11, 15, 16):*

1 3 5 6 9 ● ● 14 ● ● 18 19 20 21 22 25 26 27 28 29 30 31 32 33

1. Instance variable declaration

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

2. Directly invokes a constructor

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

3. Overloaded method signature

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

4. Constructor signature

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

5. Primitive type variable declaration

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

6. Reference type variable declaration

1 3 5 6 9 10 11 14 15 16 18 19 20 21 22 25 26 27 28 29 30 31 32 33

Question 2	**Performance and Analysis of Algorithms**	**12** points

For each of the following algorithms from our programming assignments, estimate the average case running time as a function of the input size ***N***.

1. *From N-Body:* calculate and sum all the forces between pairs among ***N*** bodies, then calculate their velocities, then update their positions, and then draw the bodies **at single time step *t***.

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

2. *From Conjunction Function:* merge (concatenate) two audio clips, each represented as an array of length ***N*** samples.

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

3. *From Recursive Graphics:* draw the Sierpinski triangle using ***N*** recursive levels, assuming that drawing a single line segment takes constant time.

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

4. *From Guitar Hero:* insert or remove one item in a `RingBuffer` object, represented as an array containing ***N*** items. (Recall that a `RingBuffer` stores the array indices of the first and last elements.)

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

5. *From Markov Model:* insert a single k-gram into an **`ST`** object containing ***N*** k-grams.

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

6. *From TSP:* insert ***N*** cities into a TSP tour using the nearest insertion heuristic.

1	$\log N$	N	$N\log N$	N^2	N^3	2^N	3^N	$N!$
○	○	○	○	○	○	○	○	○

This reference card may be useful for the following problem (Question 3).

```
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
```

Question 3 — Number Representation and TOY — **16** points

Number representation: Suppose that you have a 16-bit computer word, using **two's-complement** representation for integers. In the spaces to the right, write the **4-digit hexadecimal** representation of each entity described on the left.

4 hex digits, one per box

1. Decimal **`50`**

2. Decimal **`-50`**

3. Decimal **`-0`**
(i.e., negative zero)

4. The maximum integer

TOY: Consider what happens when the following TOY program is executed - assume the program counter is initially set to memory address `10`:

```
01: 0003   constant 0x0003
02: 0002   constant 0x0002
03: 0001   constant 0x0001
10: 7103   R[1] <- 0003
11: 8210   R[2] <- M[10]
12: 1212   R[2] <- R[1] + R[2]
13: 9214   M[14] <- R[2]
14: 0000   halt
15: 0000   halt
16: 0000   halt
```

4 hex digits, one per box

5. What is the value of **`R[1]`** immediately after the instruction at address **`10`** completes?

6. What is the value of **`R[2]`** immediately after the instruction at address **`11`** completes?

7. What is the value of **`M[14]`** immediately after the instruction at address **`13`** completes?

8. What is the value of **`R[1]`** when the program **halts**?

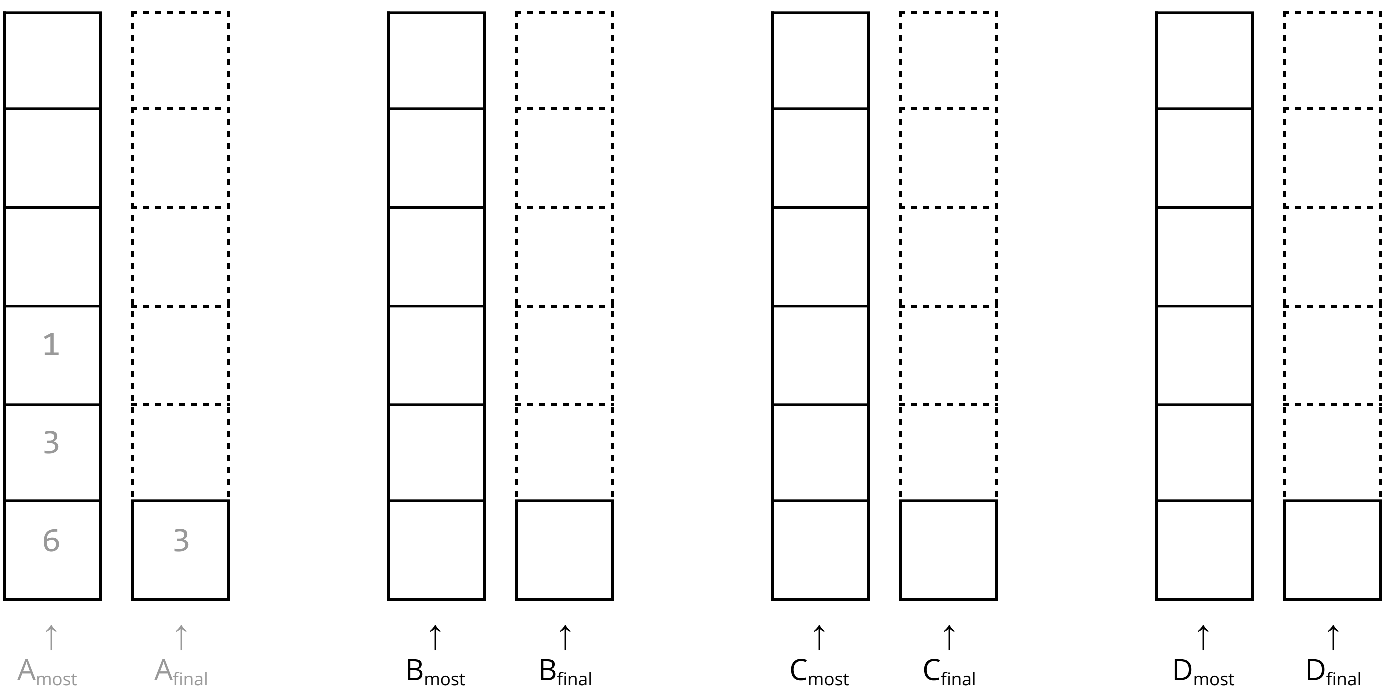
Question 4 — **Data Structures** — **12** points

Recall from the *Stacks* lecture that in **postfix** notation a binary operator (like `+`, `–`, `/` or `*`) sits after a pair of numbers on which it operates. For example, in postfix notation, the expression `(3 * 2) / (3 – 1)` would be written as: `3 2 * 3 1 – /` . You can use a stack to evaluate a postfix expression, by scanning it from left to right:

1. When you see a number *n*, **push** it on the stack.
2. When you see an operator *op*:
    a. **Pop** number n_1 off the stack.
    b. **Pop** number n_2 off the stack.
    c. Calculate (n_2 *op* n_1). For example, `(3 * 2) = 6`.
    d. **Push** the calculated result (6) on the stack.
3. At the end, the final result remains on the "top" of the stack.

Use this approach to evaluate the postfix expressions (A-D). For each, show the contents of the stack when it is most full, and when it contains the final evaluated result. The solution for the first expression, A, is provided as an example (in gray). *Hint: use scratch paper to work out your solution, and then write your final answers in the boxes below.*

A. `3 2 * 3 1 – /`
B. `3 2 – 3 1 / *`
C. `1 1 1 1 1 – + – +`
D. `1 2 3 + 4 5 * * +`

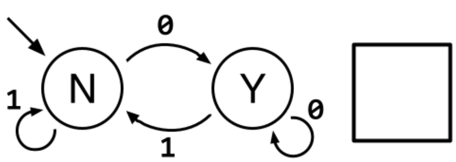


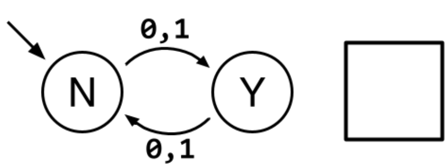
Question 5 — **REs and DFAs** — **15** points

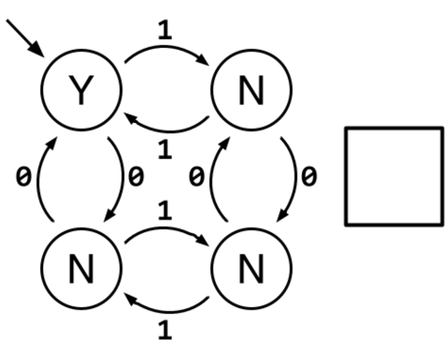
1. Assume the alphabet is {a, b, d, e}. Next to each of the RE's below, mark circles in the columns corresponding to **all** strings matched by each RE.

REs	add	baba	babe	bad	bed	dabbed
`b.*`	○	○	○	○	○	○
`b.b.`	○	○	○	○	○	○
`...`	○	○	○	○	○	○
`.*ab.*`	○	○	○	○	○	○
`.*(b|d)(b|d).*`	○	○	○	○	○	○
`(a|b|d).*d`	○	○	○	○	○	○

2. For each of the following three DFAs, write the letter (A-I, in the square to its right) that corresponds to the description that best specifies the set of strings that the DFA accepts.







Descriptions:

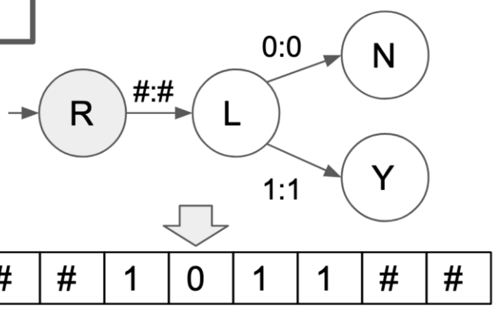
A. Any binary string.
B. Any binary string starting with `0`.
C. Any binary string ending with `0`.
D. Any binary string with even length.
E. Any binary string with odd length.
F. Any binary string with equal numbers of `0`'s and `1`'s.
G. Any binary string with an even number of `0`'s and an even number of `1`'s.
H. Any binary string that is a palindrome (same forwards and backwards).
I. Any binary string representing a number that modulo four is equal to zero.

Question 6 — **Turing Machines** — **12** points

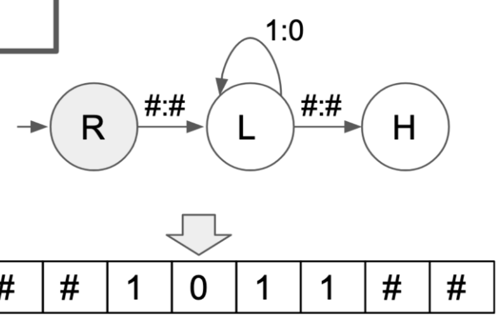
In the square above each Turing Machine, write the letter corresponding to the most accurate description of what it computes. Assume the tape starts with a valid **4-bit two's complement binary number**, and the arrow shows the starting position of the tape head.

A. Recognize if the number is negative.
B. Recognize if the number is positive.
C. Recognize if the number is odd.
D. Recognize if the number is even.
E. Add 1 to the number.
F. Negate the number.
G. Set the number to 0.
H. Set the number to -1.

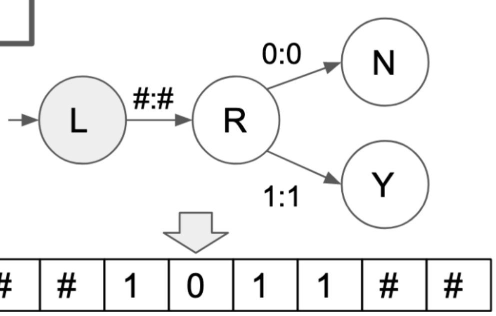
1. ☐



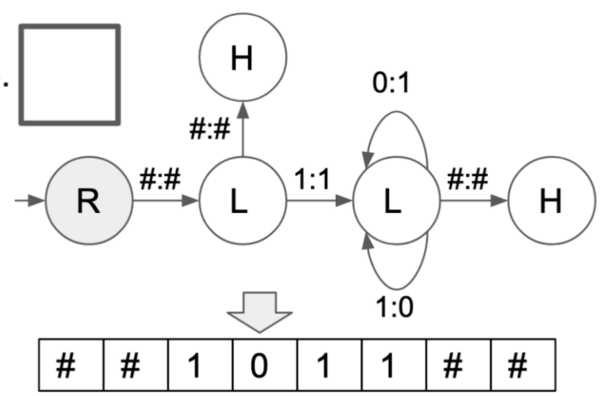
2. ☐



3. ☐



4. ☐



Question 7	**Theory of Computing**	**10** points

For each statement, select **one** of:

- **T** for TRUE,
- **F** for FALSE,
- **?** for *nobody knows for sure,* or
- **X** for *I don't know for sure.*
  If you choose **X** you will receive partial credit (0.4 point, as opposed to 1 point for correct).

	T	F	?	X	Statement
1.	◯	◯	◯	◯	P ≠ NP
2.	◯	◯	◯	◯	FACTOR is not in P. Therefore, the RSA encryption system is secure.
3.	◯	◯	◯	◯	FACTOR is something that a Turing Machine could solve, given sufficient time.
4.	◯	◯	◯	◯	If P equals NP, then finding an optimal traveling salesperson (TSP) tour can be performed in polynomial time by a TOY Machine with sufficient memory.
5.	◯	◯	◯	◯	An optimal TSP tour can be computed in polynomial time for any input of size N.
6.	◯	◯	◯	◯	Because the Halting Problem is unsolvable, even a clever instructor cannot write an auto-grader that determines in *all* cases whether a student solution for an assignment has an infinite loop.
7.	◯	◯	◯	◯	A 2022 iPhone 13 can solve some problems in NP that could not be solved by a Universal Turing Machine.
8.	◯	◯	◯	◯	An algorithm that yields solutions to arbitrary 3-SAT inputs must be intractable.
9.	◯	◯	◯	◯	There exists a Turing Machine that can simulate another specific Turing Machine running on a specific input.
10.	◯	◯	◯	◯	All problems in NP reduce efficiently (i.e., in poly-time) to NP-Complete problems.

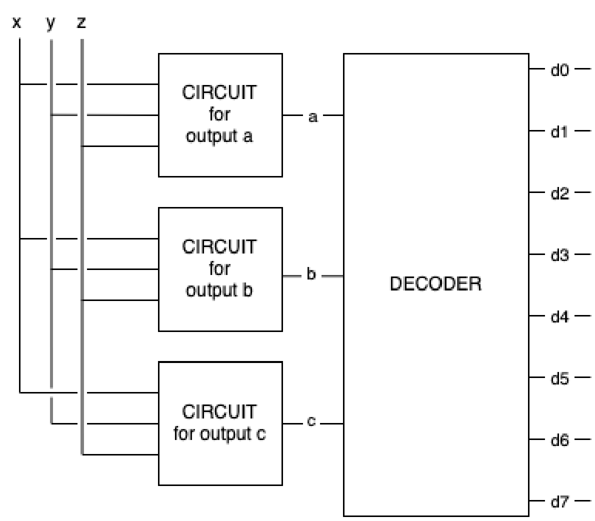
Question 8 — **Combinational Circuits** — **10** points

1. Suppose you are designing three combinational circuits with outputs (**a**, **b**, **c**), sharing the same three inputs (**x**, **y**, **z**), as specified by the truth table on the right. The boolean functions determining the values of (**a**, **b**, **c**) can be expressed as sums-of-products. Select **all** the terms that are part of the sum-of-products formula for each output below (**a**, **b**, **c**), or NONE if there are none.

Inputs			Outputs		
x	**y**	**z**	**a**	**b**	**c**
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	1

Output	**NONE**	**x'y'z'**	**x'y'z**	**x'yz'**	**x'yz**	**xy'z'**	**xy'z**	**xyz'**	**xyz**
a	○	○	○	○	○	○	○	○	○
b	○	○	○	○	○	○	○	○	○
c	○	○	○	○	○	○	○	○	○

2. The three circuits (**a**, **b**, **c**) are connected to a 3-8 decoder, where bit **a** is the most significant bit:



Of the decoder output lines **d0, d1, d2, ..., d7**, mark the ones that could possibly output a **1**:

d0	**d1**	**d2**	**d3**	**d4**	**d5**	**d6**	**d7**
○	○	○	○	○	○	○	○

3. The circuit drawn above takes inputs **x, y, z** and outputs **d0, d1, d2, ..., d7**. Which single word best describes the function of that circuit?

decrement	*count*	*shift*	*negate*	*majority*	*parity*	*difference*
○	○	○	○	○	○	○