COS126 **Written Exam 2** **Spring** '25

Instructions. This exam has eight (8) questions worth a total of one hundred (100) points. You have 80 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 bubbles and checkboxes completely: ⚫ (not ✔ or ✘). To change an answer, erase it completely and redo.

Resources. The exam is closed book, except that you are allowed to use a single 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, **write and sign the Honor Code pledge**. You may enter this information now. Again, please write neatly and legibly.

NAME:

SPRING 25 WE2

NETID (not email alias):

PRECEPT:

EXAM ROOM: ◯ McCosh 50 ◯ McCosh 46 ◯ McCosh 28 ◯ OTHER _____

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

SIGNATURE:

Question 1	**Java**	**15** points

Fill in the bubbles indicating whether each statement below is either True or False.

True	False	
◯	◯	On page 1: I have properly written my name and precept, and written and signed the honor code pledge.
◯	◯	Method overloading occurs when a class defines multiple methods with the same name but different ordered lists of parameter types.
◯	◯	In Java, an instance method of object **A** that is passed an argument referencing another object **B** of the same class can directly access private variables of **B**.
◯	◯	The keyword **`final`** applied to a reference variable prevents the object it references from being modified.
◯	◯	When creating a custom **`IllegalArgumentException`**, you can include in the exception message the value of the illegal argument in order to help debugging.
◯	◯	The Java **`String`** data type is immutable.
◯	◯	The Java **`StringBuilder`** data type is immutable.
◯	◯	Data type encapsulation reduces memory overhead in large programs.
◯	◯	Data type encapsulation protects data integrity by controlling access.
◯	◯	The **`==`** operator can always be used to determine whether two **`String`** objects have the same sequence of characters in Java.
◯	◯	In Java, all constructors must declare a return type.
◯	◯	A static method can directly access non-static variables of its class in Java.
◯	◯	Variables that are declared within a code block enclosed in curly brackets like **`{code block}`** are accessible any time after that block is executed.
◯	◯	An enhanced for loop can be used to iterate over **`Stack`**, **`Queue`**, and **`ST`** objects, but may not be used to iterate over arrays.
◯	◯	In Java, adding two huge positive integers that would overflow will result in an exception.

Question 2	**Analysis of Algorithms**	**10** points

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

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

Constant (1)	*Logarithmic (log N)*	*Linear (N)*	*linearithmic (NlogN)*	*Quadratic (N^2)*	*Cubic (N^3)*	*Exponential (c^N)*
◯	◯	◯	◯	◯	◯	◯

2. *From Hamming:* decode a sequence of ***N*** bits.

Constant (1)	*Logarithmic (log N)*	*Linear (N)*	*linearithmic (NlogN)*	*Quadratic (N^2)*	*Cubic (N^3)*	*Exponential (c^N)*
◯	◯	◯	◯	◯	◯	◯

3. *From Perceptron:* invoke the Perceptron constructor (**`new Perceptron(N)`**), represented as a symbol table **`ST<Integer, Double>`** (instead of an array of ***N*** doubles for the weights).

Constant (1)	*Logarithmic (log N)*	*Linear (N)*	*linearithmic (NlogN)*	*Quadratic (N^2)*	*Cubic (N^3)*	*Exponential (c^N)*
◯	◯	◯	◯	◯	◯	◯

4. *From Guitar Hero:* insert or remove one item in a **`RingBuffer`** object, represented as a **`Queue<Double>`** (instead of an array of N **`double`** values, as in the assignment).

Constant (1)	*Logarithmic (log N)*	*Linear (N)*	*linearithmic (NlogN)*	*Quadratic (N^2)*	*Cubic (N^3)*	*Exponential (c^N)*
◯	◯	◯	◯	◯	◯	◯

5. *From Chat126:* insert a single k-gram into an **`ST`** object containing ***N*** k-grams.

Constant (1)	*Logarithmic (log N)*	*Linear (N)*	*linearithmic (NlogN)*	*Quadratic (N^2)*	*Cubic (N^3)*	*Exponential (c^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

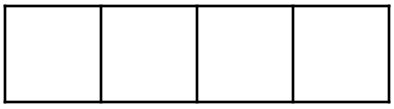
Question 3	**Number Representations and TOY**	**13** points

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

4 bits, one per box

1. Decimal **5**

2. Decimal **–5**



3. The largest integer

4. The smallest integer that does not change value when negated.

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**?

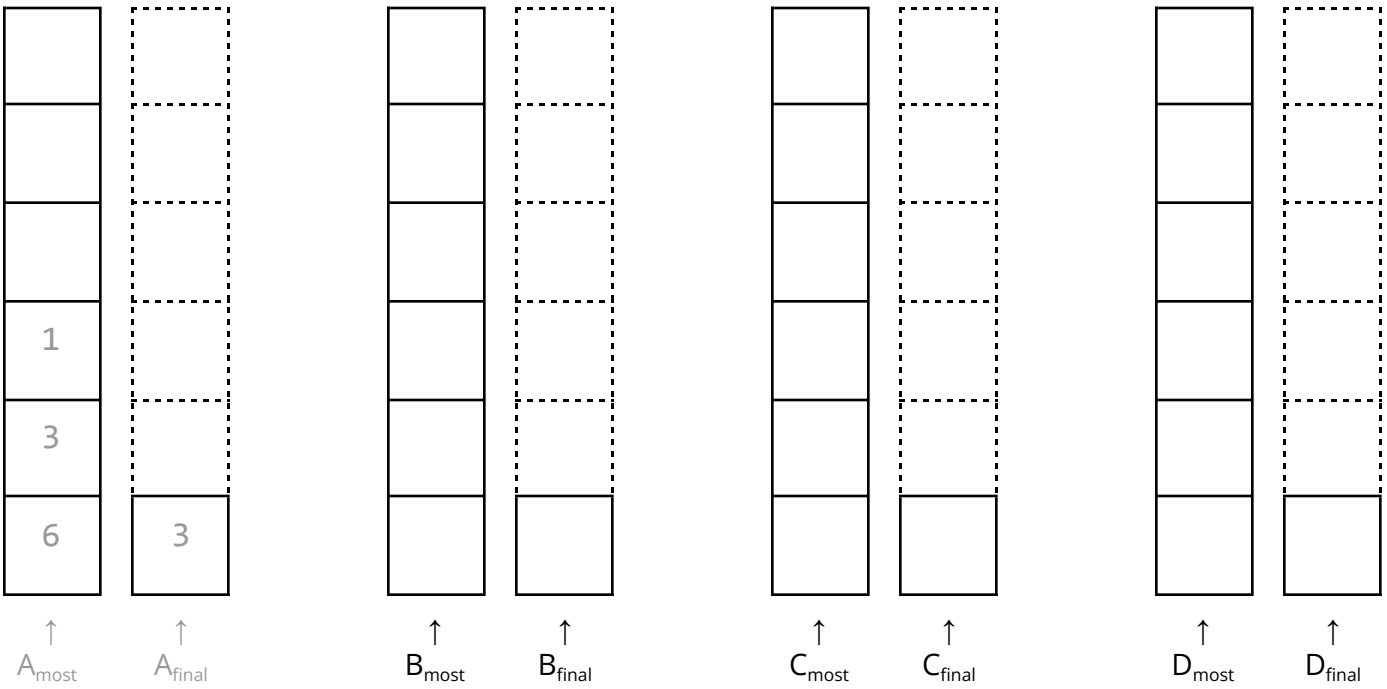
Question 4	**Data Structures**	**12** points

In **postfix** notation a binary operator (like +, –, / or *) sits after a pair of numbers on which it operates. For example this infix expression: **`(3 * 2) / (3 – 1)`** is written in postfix 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 * * +`

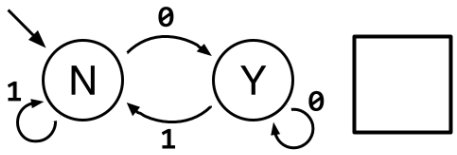


Question 5	**DFAs and Turing Machines**	**12** points

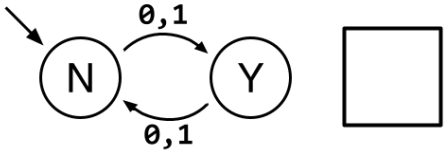
1. Mark True or False for the following statements about DFAs and Turing Machines:

True	False	
◯	◯	A DFA can recognize the language of all strings with an equal number of P's and Q's.
◯	◯	Every language recognized by a DFA can also be recognized by **both** a Turing machine and a Java program.
◯	◯	Turing machines can solve all computable problems.
◯	◯	A Turing machine with multiple tapes can solve problems that cannot be solved by one with only a single tape.
◯	◯	There exist problems that cannot be solved by any Turing machine.
◯	◯	A universal Turing machine can simulate any other Turing machine.

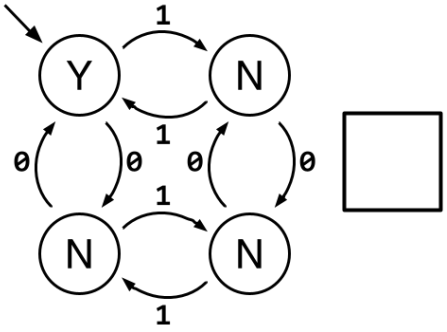
2. For each of the following three DFAs, write the CAPITAL letter (A-Z, in the square to its right) that corresponds to the description that best specifies the set of strings that the DFA accepts.



☐



☐



☐

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.

W. Any binary string with equal numbers of `0`'s and `1`'s.

X. Any binary string with an even number of `0`'s and an even number of `1`'s.

Y. Any binary string that is a palindrome (same forwards and backwards).

Z. Any binary string representing a number that modulo four is equal to zero.

Question 6	**Objects**	**18** points

A class **`Sensor`** tracks a digital sensor's activation state using the following API:

- **`Sensor(String name)`** creates a Sensor, default state is **off**.
- **`void activate()`** turns the sensor **on**.
- **`void deactivate()`** turns the sensor **off**.
- **`boolean isActive()`** returns **`true`** if the sensor is **on**, or **`false`** if the sensor is **off**.

Consider the following partial implementation of a data type that represents a *collection* of sensors, with:

- three alternative implementations for the constructor **`(A,B,C)`** and
- three alternative fragments used during testing **`(X,Y,Z)`**.

```
public class SensorList {

    private Sensor[] sensors;

    public SensorList(Sensor[] sArray) {

        // MISSING CODE FRAGMENT A, B or C          ←

    }

    public int activeCount() {
        int count = 0;
        for (int i = 0; i < sensors.length; i++) {
            if (sensors[i].isActive())
                count++;
        }
        return count;
    }


    public static void main(String[] args) {
        Sensor temp  = new Sensor("temperature");
        Sensor humid = new Sensor("humidity");
        Sensor press = new Sensor("pressure");
        Sensor light = new Sensor("light");
        Sensor[] devices = { temp, humid, press, light };
        SensorList network = new SensorList(devices);

        // MISSING CODE FRAGMENT X, Y, or Z         ←

        StdOut.println(network.activeCount());
    }
}
```

```
// Fragment A
sensors = sArray;

// Fragment B
int n = sArray.length;
sensors = new Sensor[n];
for (int i = 0; i < n; i++)
    sensors[i] = sArray[i];

// Fragment C
int n = sArray.length;
Sensor[] sensors = new Sensor[n];
for (int i = 0; i < n; i++)
    sensors[i] = sArray[i];
```

```
// Fragment X
temp.activate();
humid.activate();

// Fragment Y
temp.activate();
temp = humid;
temp.deactivate();

// Fragment Z
temp.activate();
humid.activate();
light.activate();
devices[3] = devices[2];
```

Question 6 continued...

For each combination of code fragments below, write the output of the program in the corresponding box.
If the program has a compile-time or run-time error, write the CAPITAL letter **E** in the box.
The empty space on the page may be used for scratch work.

☐ **A,X**

☐ **A,Y**

☐ **A,Z**

☐ **B,Y**

☐ **B,Z**

☐ **C,Z**

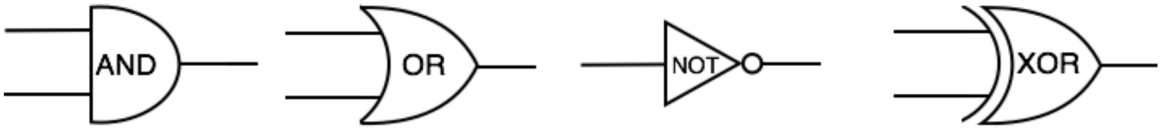
Question 7	**Machine Learning**	**10** points

True False

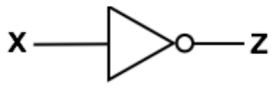
- ◯ ◯ A perceptron is a simplified model of a biological neuron that takes a vector of real numbers as input and outputs either a +1 or -1 binary label.
- ◯ ◯ The weighted sum calculation in a perceptron is essentially computing the dot product between the input vector and the weight vector.
- ◯ ◯ The perceptron algorithm makes a single pass over all training data of known input-output pairs to form the model.
- ◯ ◯ In a multiclass classification problem with m classes, we create m perceptrons, each solving its own binary classification problem.
- ◯ ◯ The test error rate is the fraction of testing inputs on which the algorithm produces false positives.
- ◯ ◯ The `extractFeatures()` method in *ImageClassifier* converts a 2D image of width w and height h into a 1D array with length equal to $(w^2 + h^2)$.
- ◯ ◯ When using a supervised learning algorithm, the training and testing phases use the same data.
- ◯ ◯ If a training dataset is imbalanced (contains more examples of some classes than others), perceptrons will generally perform better on the more frequent classes but may show bias against less represented classes.
- ◯ ◯ The geometrical interpretation of the perceptron algorithm is that it attempts to find a hyperplane that perfectly separates positive and negative examples, and each weight update moves the hyperplane in the direction that correctly classifies the misclassified example.
- ◯ ◯ Unlike supervised and unsupervised learning, reinforcement learning always requires a simulated environment for the agent to interact with.

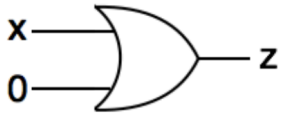
Question 8	**Circuits**	**10** points

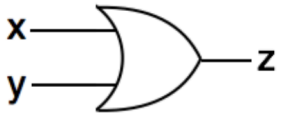
First, a reminder of the shape of the following logic gates:

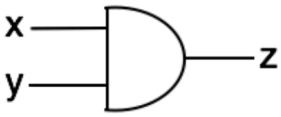


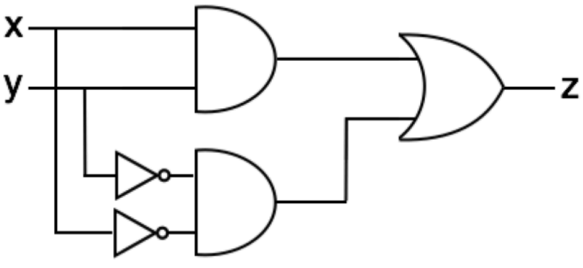
In the left column below, there are circuits that use only **AND**, **OR**, and **NOT** gates.
In the right column below, there are circuits that use only **XOR** gates.
Fill in the boxes on the left with the CAPITAL letter of the equivalent circuit on the right.
If **no** circuit on the right matches, write the CAPITAL letter **N** in the box.

☐ 

☐ 

☐ 

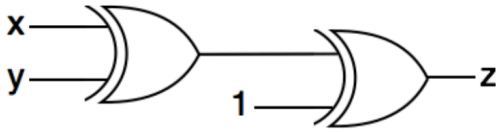
☐ 

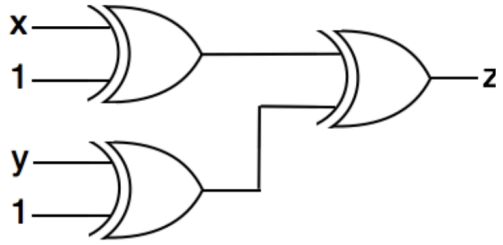
☐ 

A 

B 

C 

D 

E 

N (no matching circuit using **XOR** gates)

THIS PAGE INTENTIONALLY EMPTY