Software Developer - Technical Assignment

Please complete the assignments below in the language of your choice. Put the solution for each assignment in a separate folder, organize your code at a production level and make sure your code is well tested. Please refer to the provided rubric we use to evaluate responses to this assignment.

Assignment 1

We are evaluating a binary classification model. We have the number of true positives, true negatives, false positives, and false negatives for confidence score thresholds 0.1, 0.2, 0.3, ..., 0.9 respectively (feel free to assume the data structure for this input data and provide an explanation).

Write a function to return THE BEST threshold that yields a recall >= 0.9. Unit tests for this function are also encouraged.

(Note that you don't have to write unit tests for this function. Alternatively, you must provide example code for calling the function with realistic input data.)

Assignment 2

Objective

This assignment tests your Object-Oriented Design skills. Consider a string of ones and zeros representing an unsigned binary integer. Design and implement a solution that will compute the remainder when the represented value is divided by three.

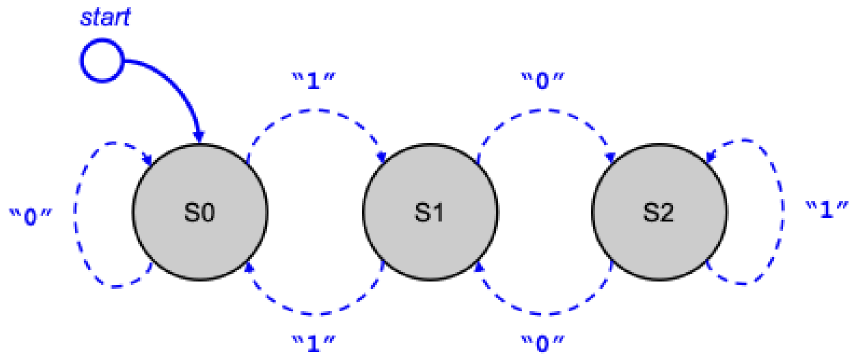
For example:

Input: '1101' Output: 1
Input: '1110' Output: 2
Input: '1111' Output: 0

One way to implement this would be to convert the input string to a number type and use the modulus operator (%). While that approach will produce the correct answer, for this exercise we ask you to use a more interesting method: Finite State Machine (FSM).

What is Finite State Machine (FSM)?

Let us solve this mod-three problem by a generic FSM. It takes the input characters, one at a time, MOST significant bit first and transitions between three states: S0, S1, S2.



The value returned from our function will depend on the state selected after the character sequence is exhausted. The final state will be converted to a remainder value as specified in the following table:

Final State	Remainder
S0	0
S1	1
S2	2

For input string `"110"`, the machine will operate as follows:

1. Initial state = S0, Input = 1, result state = S1
2. Current state = S1, Input = 1, result state = S0
3. Current state = S0, Input = 0, result state = S0
4. No more input – return the remainder value corresponding to the final state S0.

For input string `"1010"` the machine will operate as follows:

1. Initial state = S0, Input = 1, result state = S1
2. Current state = S1, Input = 0, result state = S2
3. Current state = S2, Input = 1, result state = S2
4. Current state = S2, Input = 0, result state = S1
5. No more input - return the remainder value corresponding to the final state S1.

FSM Implementation

The FSM described above is an instance of finite state automata. With the abstraction provided below, use an **object-oriented design (OOD)** approach to design and implement a **generic FSM class**. The interfaces of your class should be designed for use by other developers for any FSM problems, not just mod-three problem. Lastly write the code to solve the mod-three problem with the FSM class - either a subclass or a function would work.

Finite Automation

A finite automaton (FA) is a 5-tuple (Q, Σ, q0, F, δ), where

- Q is a finite set of states;
- Σ is a finite input alphabet;
- q0 ∈ Q is the initial state;
- F ⊆ Q is the set of accepting/final states; and
- δ: Q×Σ→Q is the transition function.

For any element q of Q and any symbol σ ∈ Σ, we interpret δ (q, σ) as the state to which the FA moves, if it is in state q and receives the input σ.

Mod-Three FA

Based on the notation from the definition, the modulo three FSM would be configured as follows:

- Q = (S0, S1, S2)
- Σ = (0, 1)
- q0 = S0
- F = (S0, S1, S2)
- δ(S0,0) = S0; δ(S0,1) = S1; δ(S1,0) = S2; δ(S1,1) = S0; δ(S2,0) = S1; δ(S2,1) = S2