Assignment 2

Due Date: November 13, 2023 at 12:30 PM

Overview

This assignment will be worth **9%** of your total grade. The goal is to understand, test, and deploy local smart contracts using hardhat.

Specifications

Part I - 75 points

Create a college enrollment smart contract named "CSEnrollment" for the computer science department at Towson. For this enrollment system, you will have four total CS courses on initialization:

- 670 (graduate)
- 617 (graduate)
- 484 (undergraduate)
- 431 (undergraduate)

The enrollment system will store information for all the students who have enrolled for a course. After creating the contract, we should add the owner of the enrollment system. Below are the minimum function requirements for the enrollment contract:

- *register* (credits, studentType, course): save when a course is registered by a student. A student can only register for **one** course. Undergraduates can only register for 484 and 431. Graduate students can register for any course, only if they have over 20 credits, but they are not allowed to register for 431. No class can have more than 30 students registered at a given time.
  - credits: number
  - studentType: either graduate or undergraduate
  - course: either 670, 617, 484, and 431 (this can be a string, number, or something else depending on how your contract is set up)
- *add* (courseNumber, courseType): only the owner can add a new course to the enrollment contract. The owner cannot add a course that already exists (same courseNumber)

  - courseNumber: course number (i.e. 670)
  - courseType: either graduate or undergraduate

- *getRoster* (course): log the student addresses enrolled for a given course
- **OPTIONAL:** *drop* (course): student can drop a course as long as it has been less than 30min since registration

Feel free to add appropriate events for the required functions or to add extra functions to enhance your contract. A few notes/hints:

- It might be useful to make a student struct. You are not limited to what you can store for a user
- You do not need to worry about a student registering for a course that was added by the owner after contract initialization
- **OPTIONAL:** Time in Solidity is simple, check here for a reference

Part II - 25 points

For your a2p2.js (your run.js), include testing for each function using the outline below:

- student registering for a course (successful)
- student registering for a course (unsuccessful)
- owner adding a course (successful)
- owner adding a course (unsuccessful)
- getting the roster for a course that exists with 0 students
- getting the roster for a course that exists with more than 0 students
- getting the roster for a course that does not exist
- **OPTIONAL:** student dropping a registered course (successful)
- **OPTIONAL:** student dropping a registered course (unsuccessful)

Note: you may need to call getRoster multiple times to show a successful or unsuccessful registration and drop. The goal is to show the functionality of the contract exists.

Submission

Please submit the following on Blackboard (you may need to .zip the contents into a single folder):

- a2p1.sol
- a2p2.js