Free University of Bolzano Bozen – Faculty of Economics and Management

Big Data and Blockchain exam

# Rules

* + No communication with other people or among students is allowed. Phones and every other means of communication must be turned off. Opening any communication program on the computer is not allowed and is considered cheating.
  + You are responsible for the correct copy of your files.

## Exercise Smart Contracts

You have 70 minutes starting from now.

## Idea

Your task is to build a contract to handle certification. The applicant must submit an enrolment and pay a fee. After an exam is passed, the certification authority (which is the deployer of the contract) inserts the certification’s data: name, level, time. Anybody can provide an address, name and level and get the information whether that person has at least that certification’s level and when it was taken. The user has always the right to delete his/her own certification, paying the fee.

## Task

Write Solidity contract **certification**.

## Suggested data structures

    mapping(address=>uint) private application\_time;

    mapping(address=>string) private certification\_name;

    mapping(address=>uint) private certification\_time;

    mapping(address=>uint) private certification\_level;

    address payable public authority;

    uint public fee;

## Step 1

Build a **constructor** which sets authority and fee.

## Step 2

Add a function **enrol**. The user must pay at least the fee. Function sets the current time as application time. He/she does not need to specify his/her name, nor the requested level.

**FLIP PAGE 🡪**

## Step 3

Add a function **insert**, which is usable only by the authority. The authority passes as parameter an address, a name and a level. If that address already has a higher certification or if the application’s time is older than 90 days, the function returns **false**. Otherwise, the function updates all the mappings, resets the application’s time and returns **true**.

## Step 4

Add a view **check** which takes as parameters an address, a name and a level. If that address corresponds to the provided name and has a certification’s level of at least that level, it returns the certification’s time. Otherwise it returns 0.

Attention: when you try to compare two strings, the compiler gives you an error *TypeError: Built-in binary operator == cannot be applied to types string storage ref and string memory*. The solution is to compare two strings not with STRING1 == STRING2 but with  
keccak256(abi.encodePacked( STRING1 )) == keccak256(abi.encodePacked( STRING2 )).

In case feel free to call me during the exam.

## Step 5

Add function **cancel** through which a user cancels all his/her data (string variables are set to “”). The fee must be paid in order for the function to work.

## Step 6

Deploy the contract on Sepolia’s blockchain and test all the functions using Account 1 for the authority and Account 2 for the applicant.

If you did not manage to do it fully, after having copied everything into a text file, remove the problematic functions and deploy and test what you have.

## Save and return:

* a text file **certification.txt** containing the Solidity source code that you have built and your Sepolia’s contract’s address