Lab 1 – Build a Blockchain in Python

Information Security UESTC4036_2023, School of Engineering, University of Glasgow

Student Name: ____

Student Number: ____

UESTC4036_2023 Lab Assessment

Each individual lab will be evaluated using the below criteria:

- Design a blockchain system in Python– 50%

  This means each student will need to demonstrate a functional blockchain system for each task in the lab manual to the GTA and member of staff to get the full mark.

- Lab Questions – 10%
- Reflection and significance – 25%

  After the completion of programming and testing each of the “Tasks”, you will be required to reflect, in a few lines, on your understanding of the tasks and its significance. This is a chance for you to ask yourselves “What you did?”, “What it means?”, and “What next?”, meaning what is the significance of the undertaken task in the context of blockchain and what you are learning in the lecture. Try to be innovative and think outside the box. Please, do not provide a summary of what you did as it will not be accepted.

- Adherence to coding good practice – 15%
  - Meaningful variable and function names – please avoid names like “x”, “y”, or personal names
  - Using annotate comments to explain what your code does at the beginning and to clarify what each important line of code does.
  - Consistent indentation – Indentation helps in making the code readable and to avoid error.

1. Lab 1 Objective

The objective of this lab is to design and test blockchain technology and understand its components in detail. A Proof of Work (PoW) consensus is used in this blockchain. In this lab, we will focus more on the implementation part.

2. Software Requirements

Software:

- **Python**
- **Visual Studio Code**

3. Task 1: Python Tutorial

Install Python and Visual Studio Code

Access to the webpage *https://docs.python.org/3/tutorial/* to get more information about Python coding.

Task1.1: Create an empty list and use a for loop to generate 10 blocks and append these blocks into the list. For each blockchain, the datatype is *dict*, there are index and data in equal steps starting from 100 until it reaches 1000. Data need convert to list data type.

Task1.2: Define a function to achieve the block record in task3.1, and write another function to print the block according to the input index.

Task1.3: Put both functions into a class. Write a main program which creates an instance of the class and can print desired block.

Task1.4: Comment your code, not just with inline comments (beginning '#'), but write a docstring comment (one which is enclosed in triple quotes) for the function, detailing what arguments the function takes and what it returns.

4. Task 2: Build The Environment

Launch the command portal and create a new document with a virtual environment.

After setting up the environment, open the IDE create a new file for coding, and name it *blockchain.py*.

In the *blockchain.py* file, first import the following packages as they are required in building our blockchain:

time (or *datetime*), *json, hashlib*

Use the *time* or *datetime* library to attach a timestamp to each block that is created or mined. The *hahshlib* will be used to hash a block, *JSON* will be used to encode the block before we hash it.

5. Task 3: Build The Blockchain Architecture

To start building the PoW blockchain, please create a *Blockchain* class. The __init__ method will consist of a variable called chain to store a *list* of all the blocks in the blockchain.

The structure of generated blockchain is as below

{

index: 0, #Genesis Block

timestep: current time,

data: "Hello World!",

proof: 0,

previous_hash: '0'

} -> hash() -> 3s2351h

{

index: 1,

timestep: current time,

data: "Transaction A",

proof: 24912,

previous_hash: 3s2351h

} -> hash() -> 8fv743413

{

index: 2,

timestep: current time,

data: "Transaction B",

proof: 235724,

previous_hash: 8fv743413

} -> hash() -> 41j566d098

Task3.1: Write a block generate function to create the Genesis block on instantiation of the class.

- Index: An index key will store the blockchain's length. It is represented by the chain variable in the __init__ method. Please use this variable to access each block in the chain.

- Timestamp: The timestamp key will take a value of the current Date and Time the block was created or mined.
- Date: The recorded information.
- Proof: This key will receive a proof value that will be passed to the function when called. Note that this variable refers to the proof of work.
- Previous hash: Lastly, the previous hash key takes a value of previous_hash from the function which is equivalent to the hash of the previous block.

Task3.2: Write a function that can input information into the block and record in blockchain.

Task3.3: Access previous block by using a simple function. Please create a method that gets the previous block in the chain.

6. Task 4: Reflection

After the completion of programming and testing please reflect, in a few lines, on your understanding of the tasks and its significance. This is a chance for you to ask yourselves “What you did?”, “What it means?”, and “What next?”, meaning what is the significance of the undertaken task in the context of blockchain system and what you are learning in the lecture. Try to be innovative and think outside the box. Please, do not provide a summary of what you did as it will not be accepted. **[Answer in the below space]**

7. Lab Score

Marking Criteria	Student’s Score
Demonstration (50%)	
Lab Questions (10%)	
Reflection (25%)	
Adherence to coding good practice (15%)	