Qatar University
College of Engineering
CMPS497 Special Topics in Computing
Blockchain and Cryptocurrencies
Spring 2020

Programing Assignment 1
Hashing and Hash Pointer Data Structures

[Graded out of 45 points]

Submission and Deadline:

- The submission deadline is **11:59 pm on February 11, 2020**
- Type your answer in this word document.
- Use blue color for your answers
- Submit a zip file on Blackboard. The file should contain the following:
    - Your typed answer sheet (this document after adding your answers)
    - Source code of the program(s) you used to answer Q4.

Throughout this assignment,

- *Assume that the cryptographic hash function used is **SHA256***
- *Replace any occurrence of **$$Name$$** with your own first name*
- *Provide all hashes in **hexadecimal**.*
- The assignment is graded out of 45 points

Q1. Hash Functions [7 points]

a. [3 points] Fill the right column of the table below by finding the hash values (digests) of each of the messages given in the left hand side of the table.

Message	SHA256 Digest
$$Name$$	
$$Name$$ will get A	
$$Name$$ will get B	

b. [4 points] Give two reasons to explain why the following hash function H is NOT considered a cryptographically secure hash function for X, where X is a large value that requires more than 256 bits.

H(X) = Most significant 256 bits of X

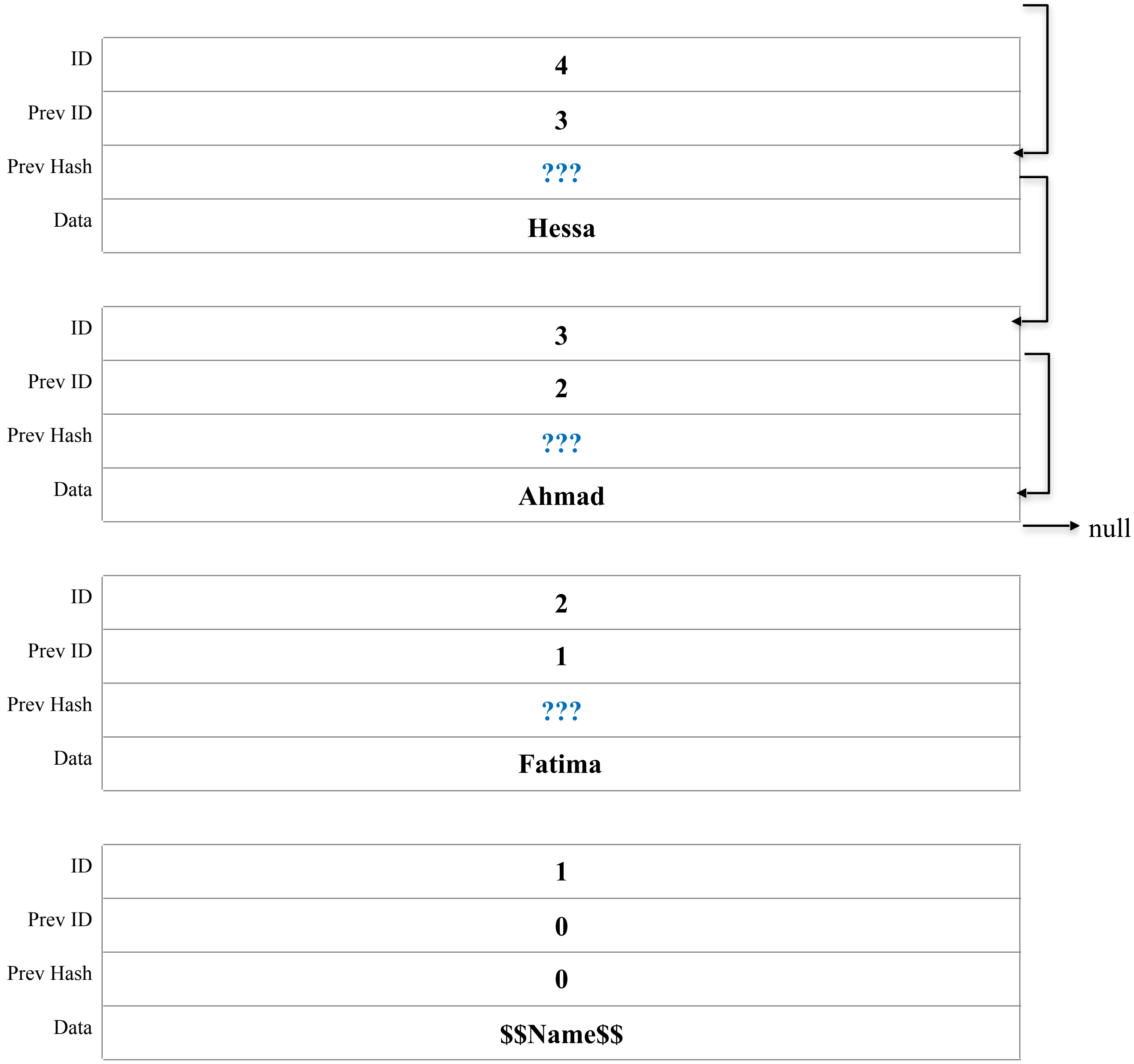
Q2. List with Hash Pointers [7 points]

The linked list below uses hash pointers. Each item in this list has three fields: ID, previous item ID, previous node hash, and data. To hash a block, the fields are concatenated in the same order. For example, to hash the block with ID=4, the following value is hashed (after replacing ??? with the right Prev Hash value): “43???Hessa”

a. [4 points] Fill the missing hash values for this list (replace ??? by the appropriate value)

Head Pointer: **4**

Head Hash: **???**



ID	**4**
Prev ID	**3**
Prev Hash	**???**
Data	**Hessa**

ID	**3**
Prev ID	**2**
Prev Hash	**???**
Data	**Ahmad**

ID	**2**
Prev ID	**1**
Prev Hash	**???**
Data	**Fatima**

ID	**1**
Prev ID	**0**
Prev Hash	**0**
Data	**$$Name$$**

b. [1 point] Suppose that you store this list on an untrusted computer. How much data you need to store to be able to detect any malicious tampering with the data?

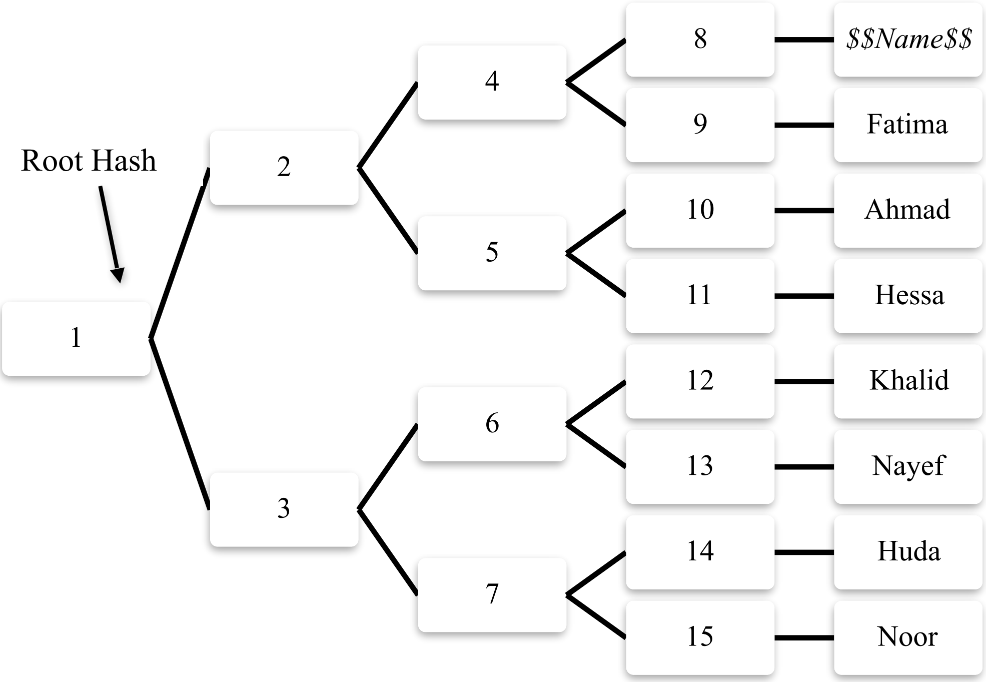
c. [1 point] Explain how you will be able to detect if someone changes “Fatima” to “Fatma”.

d. [1 point] Explain how will you be able to detect if someone changes “Fatima” to “Fatma” and also re-computes the Prev Hash values of blocks with IDs 3 and 4.

Q3. Merkle Tree [13 points]

Consider a Merkle tree shown below to store the list of 8 records: $$Name$$, Fatima, Ahmad, Hessa, Khalid, Nayef, Huda, and Noor.

a. [7 points] What is the root hash value and hash value stored in each of the shown nodes 1-7.



Node ID	**Hash value**
Root Hash	
1	
2	
4	
5	
8	

9	

b. [2 points] Which of the 16 hash values (including the root hash) of this tree need(s) to be changed by someone who is trying to tamper with the record Fatima without being detected?

c. [2 points] If you maintain the Root Hash of the tree on your computer and store the tree data on an untrusted server. The types of queries that you send the server are of the form: “Does record X exist in the database?” Whenever the servers answers “Yes” (i.e. the record exists), you want the server to also send a proof that the record exists (i.e. the server is not cheating by saying yes if the record does not exist). What would be the proof that the server needs to send (set of node hash values) when he answer the question “Does the record Fatima exist in the database?” How would we validate the answer using the proof data sent by the server?

d. [2 points] Complete the following sentence by filling the blank:

For a general Merkle tree storing N records, the size of a proof would be O(__________).

For a general Merkle tree storing N records, validating proof of membership takes O(__________) time.

Q4. Mining [18 points]

The hash target is used in mining. It is a number that a block hash must be below for the block to be valid in order for it to be added on to the blockchain. Assume that the hash target is 256=2^8. In other words, each block hash has to start with 8 zeros (as the most significant bits). To hash a block, the fields are concatenated in the same order. For example, the "Current Block Hash" for the block with ID=1 is the hash of the value "1002262Ahmad".

a. [6 points] Fill replace "???" by appropriate values to make the blockchain a valid one.

Head Hash: ???

Head Pointer: **4**

ID	**3**
Prev ID	**2**
Prev Hash	???
Nonce	???
Data	**Hessa**
Current Block Hash	???

ID	**2**
Prev ID	**1**
Prev Hash	001ecdf2e0b2b7b7c51701385d99d58eaeccbabea16061a0ef2cfb91fb308336
Nonce	???
Data	**$$Name$$**
Current Block Hash	???

→ null

ID	**1**
Prev ID	**0**
Prev Hash	**0**
Nonce	**2261**
Data	**Ahmad**
Current Block Hash	001ecdf2e0b2b7b7c51701385d99d58eaeccbabea16061a0ef2cfb91fb308336

b. [6 points] Find the average time for mining a block for each of the following hash targets. Measure the time in terms of both the number of hash computations as well as in seconds on your machine. Report your results as an average for mining 10 different blocks. Fill the table below.

   1. Hash target = 2^8
   2. Hash target = 2^{12}
   3. Hash target = 2^{16}

Hash Target	**Average number of hash computations** to mine a block (average over 10 blocks)	**Average time in seconds** to mine a block (average over 10 blocks)
$2^8 = 256$		
$2^{12} = 4{,}096$		
$2^{16} = 65{,}536$		

c. [6 points] Show the source code of a full program to mine one block. The block follows the format of the blocks shown in part (a) of this question. In the program, you initialize the values of the fields

   - ID
   - Prev ID
   - Prev Hash
   - Data

   It computes and prints the Nonce and the Current Block Hash values. You can use a programming language of your choice.

Hints

- Make sure that you stick to the block format, the order of fields, and the way of concatenating the fields. Otherwise, your answers will not match the answers that I will use to grade the assignment (as you know any change on the data will lead to completely changing the hash value, even if it is an extra space, for example).

- If you are using Java, you can use built-in **`MessageDigest`** class for SHA-256 hashing. The following shows an example of using the **`MessageDigest`** class:

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] encodedhash = digest.digest(
  originalString.getBytes(StandardCharsets.UTF_8));
```

To use `MessageDigest`, you need to include the following import statement:

```
import java.security.MessageDigest;
```

The hash result is returned in a byte array. The following code illustrates a byte to hex converter that allows you to get the hashed value as a string of hexadecimal characters:

```
private static String bytesToHex(byte[] hash) {
    StringBuffer hexString = new StringBuffer();
    for (int i = 0; i < hash.length; i++) {
       String hex = Integer.toHexString(0xff & hash[i]);
       if(hex.length() == 1) hexString.append('0');
       hexString.append(hex);
    }
    return hexString.toString();
}
```