Problem Set 1

1. Find a collision in each of the hash functions below

a. H(x) = x mod 512, where x can be any integer

Ans: The collision is x1 = 100 and x2 = 612

b. H(x) = number of 0-bits in x, where x can be any bit string

Ans: x1 = The collision is x1= “01010101" and x2 = "10101010"

c. H(x) = the five least significant bits of x, where x can be any bit string

Ans: The five least significant bits of x: Collision: 0101101and 1101101

x1 and x2 have the same five least significant bits 01101 when passed through the hash function H(x) with little endian encoding.

2. Implement a program to find an x such that H (x ◦ id) ∈ Y

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package System.out;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 *
 * @author rodri
 */
public class Hashfinder {

    public static void main(String[] args) {

        String StringId = "ED00AF5F774E4135E7746419FEB65DE8AE17D6950C95CEC3891070FBB5B03C77";
        byte[] id = hexStringToByteArray( hexString: StringId);
        try {
            MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256");
            byte[] x = md.digest( input: id);
            while (!byteValidator( bytes: x, (byte) 0xff)) {
                id[15]++;
                x = md.digest( input: id);
            }
            System.out.println("Found the value x: " + byteArrayToHexString( byteArray: x));
            System.out.println( x: "Done Finding x");

        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

    }

```

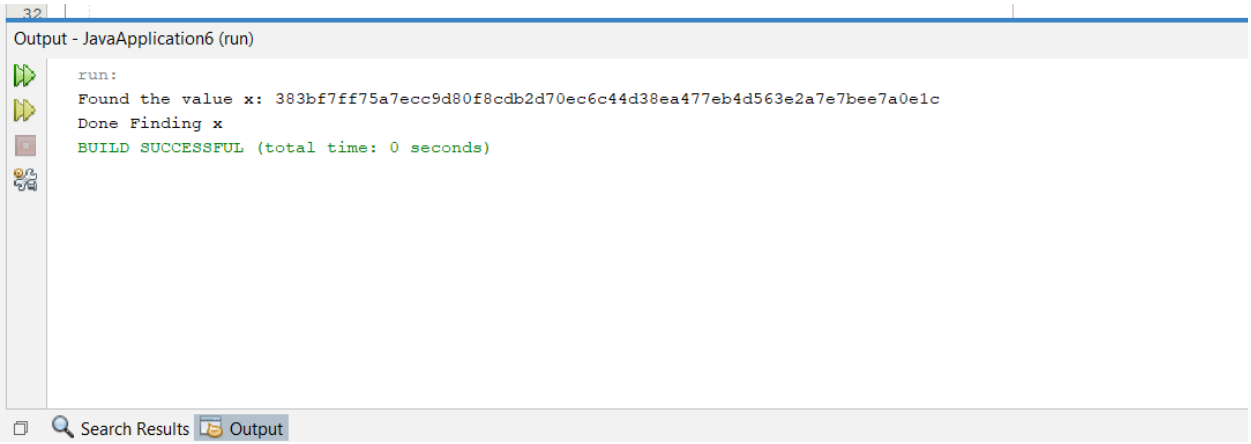
```
    private static boolean byteValidator(byte[] bytes, byte value) {
        for (byte b : bytes) {
            if (b == value) {
                return true;
            }
        }
        return false;
    }

    private static String byteArrayToHexString(byte[] byteArray) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : byteArray) {
            hexString.append( str: String.format( format: "%02x",  args: b));
        }
        return hexString.toString();
    }

    private static byte[] hexStringToByteArray(String hexString) {
        int len = hexString.length();
        byte[] byteArray = new byte[len / 2];
        for (int i = 0; i < len; i += 2) { // Updated termination condition
            byteArray[i / 2] = (byte) ((Character.digit( ch: hexString.charAt( index: i),  radix: 16) << 4)
                    + Character.digit( ch: hexString.charAt(i + 1),  radix: 16));
        }
        return byteArray;
    }

}
```

Output



3. Alice and Bob want to play a game over SMS text where Alice chooses a number between 1 and 10 in her head, and then Bob tries to guess that number. If Bob guesses correctly, he wins. Otherwise, Alice wins. However, Bob complains that the game isn't fair, because even if Bob guessed correctly, Alice could lie and claim that she chose a different number than what she initially chose. What can Alice do to prove that she didn't change the number she initially chose? Devise a mechanism to address Bob's concern. Provide a detailed explanation of the mechanism and why it works. An answer with insufficient detail will not receive credit. You should only need to use cryptographic hash functions to solve this problem. Keep the solution simple

Answer➔

To address Bob's concern and uphold the fairness of the game, Alice can employ a mechanism that incorporates a cryptographic hash function. Here's a simple solution that she can implement:

Before the game starts, Alice and Bob agree on using an SHA-256 hash function.

Steps of the mechanism

1. Alice chooses a number between 1-10 along with one letter and one Special character and keeps it in her mind.
2. Alice then calculates the hash using the agreed hash function for the given number. Let's call this value "A".
3. Alice sends hashed "A" to Bob over SMS.
4. Bob guesses a number between 1 and 10 and sends the number to Alice.
5. Upon receiving Bob’s guess, Alice reveals her chosen number.
6. Alice then calculates the SHA-256 hash for the number received from Bob. Let's call this value "B".
7. Alice compares "A" with the received "B". If the hashes match, then Bob’s guess is right, but if it’s different then bobs guess is wrong.

Here's why this mechanism works:

I. Alice's hash value, "A" acts as a commitment to her chosen number. Once Alice calculates the hash and sends it to Bob, she cannot change the chosen number without Bob detecting it.

II. When Bob reveals his guess and Alice discloses her initially chosen number, Bob can independently verify that the revealed number matches the hash value he received earlier. If Alice were to change her number, the hash would no longer match, indicating that Alice was lying.

III. Cryptographic hash functions are designed to be one-way and collision resistant. It is computationally infeasible to find a different number that matches the given hash value or to find two different numbers that produce the same hash value. Therefore, Alice cannot change her number without altering the hash value.