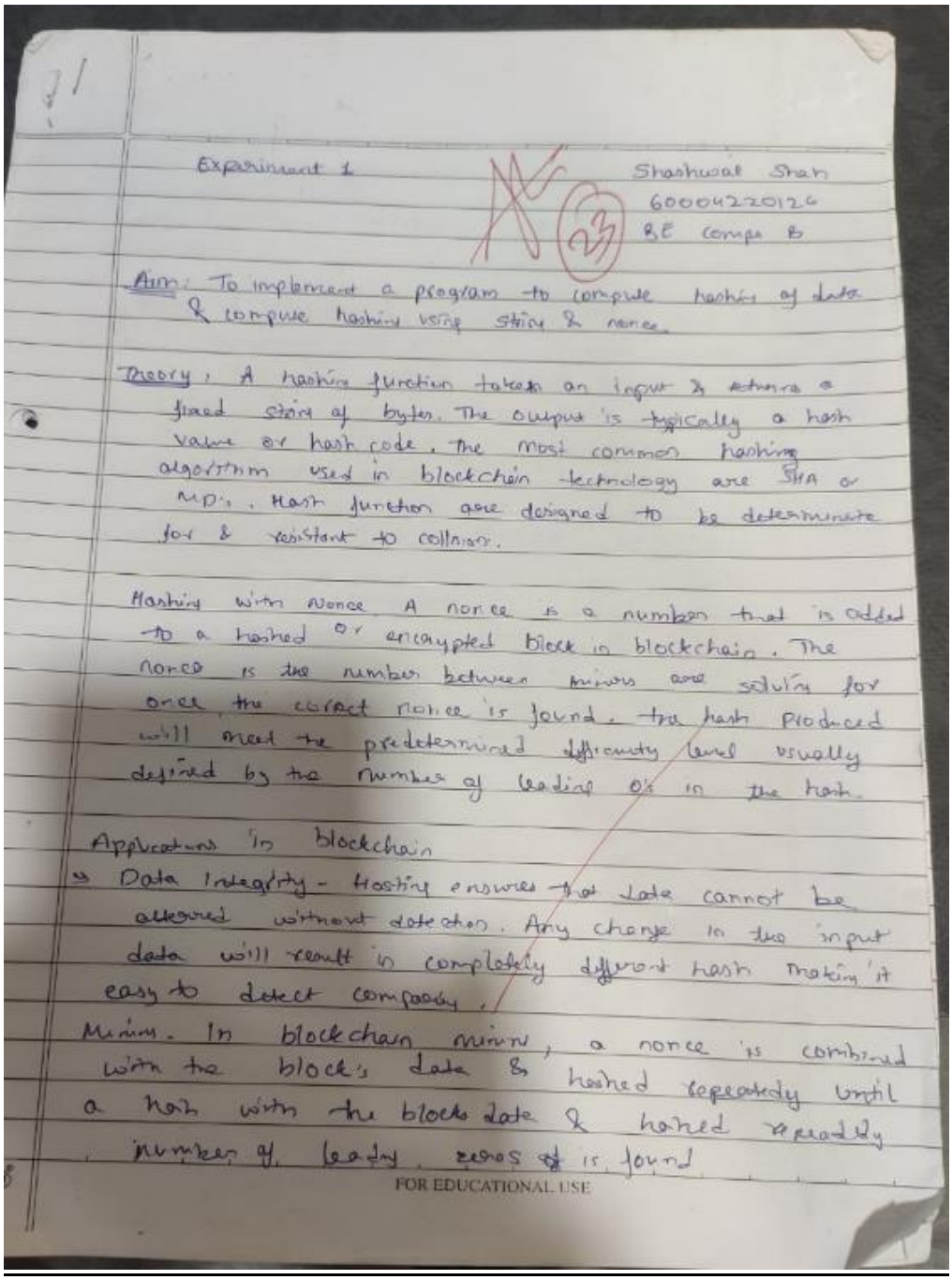
BLOCKCHAIN TECHNOLOGY

Name:-Shashwat Shah
Sapid:-60004220126
Branch:-Computer Engineering
Div/Batch:-C2-2

EXPERIMENT NO.01



Experiment 1

Shashwat Shan
60004220126
BE Comps B

Aim: To implement a program to compute hashing of data & compute hashing using string & nonce.

Theory: A hashing function takes an input & returns a fixed string of bytes. The output is typically a hash value or hash code. The most common hashing algorithm used in blockchain technology are SHA or MD5. Hash function are designed to be deterministic for & resistant to collision.

Hashing with Nonce: A nonce is a number that is added to a hashed or encrypted block in blockchain. The nonce is the number between miners are solving for once the correct nonce is found. the hash produced will meet the predetermined difficulty level usually defined by the number of leading 0's in the hash.

Applications in blockchain

» Data Integrity - Hashing ensures that data cannot be altered without detection. Any change in the input data will result in completely different hash making it easy to detect compromise.

Mining: In blockchain mining, a nonce is combined with the block's data & hashed repeatedly until a hash with the block data & hashed nonce number of leading zeros is found.

FOR EDUCATIONAL USE

CODE & OUTPUT :-

```python
import hashlib

def generate_sha256_hash(input_string):
    # Convert the input string to bytes
    input_bytes = input_string.encode('utf-8')

    # Create a SHA-256 hash object
    sha256_hash = hashlib.sha256()

    # Update the hash object with the input bytes
    sha256_hash.update(input_bytes)

    # Get the hexadecimal representation of the hash
    hash_result = sha256_hash.hexdigest()

    return hash_result

# Example usage
input_string = "Hello, Blockchain!"
hash_output = generate_sha256_hash(input_string)
print(f"Input: {input_string}")
print(f"SHA-256 Hash: {hash_output}")
```

```
Input: Hello, Blockchain!
SHA-256 Hash: 7526b1d2bc17587443fbf1fafb27e95d70615bc7576c6e34c1f139c9ce857733
```

```python
import hashlib
from PyPDF2 import PdfReader

def hash_pdf(file_path, hash_algorithm='sha256'):
    # Open the PDF file
    with open(file_path, 'rb') as file:
        # Create a PDF reader object
        pdf_reader = PdfReader(file)

        # Initialize the hash object
        hasher = hashlib.new(hash_algorithm)

        # Iterate through all pages
        for page in pdf_reader.pages:
            # Extract text from the page
            text = page.extract_text()
            # Update the hash with the text
            hasher.update(text.encode('utf-8'))

        # Return the hexadecimal representation of the hash
        return hasher.hexdigest()

# Example usage
pdf_file = 'Blockchain_Syllabus.pdf'
hash_value = hash_pdf(pdf_file)
print(f"The {hashlib.sha256.__name__} hash of the PDF content is: {hash_value}")
```

[3]

```
The openssl_sha256 hash of the PDF content is: 8bf310b94730c54d468a93df0314d52f5809284d2fc5c0610b699824adc0c7f0
```

```python
import hashlib

def compute_hash(data, nonce, algorithm='sha256'):
    # Combine the data and nonce
    combined = f"{data}{nonce}".encode('utf-8')

    # Create a hash object with the specified algorithm
    hasher = hashlib.new(algorithm)

    # Update the hasher with the combined data
    hasher.update(combined)

    # Return the hexadecimal representation of the hash
    return hasher.hexdigest()

# Example usage
data = "Hello, World!"
nonce = 0
target_prefix = "00000"  # Example: we want a hash starting with four zeros

while True:
    hash_result = compute_hash(data, nonce)
    if hash_result.startswith(target_prefix):
        print(f"Found matching hash with nonce {nonce}: {hash_result}")
        break
    nonce += 1

print(f"Final nonce: {nonce}")
print(f"Final hash: {hash_result}")
```

```
Found matching hash with nonce 1558215: 000008fb67e78dee225c2bea554b989b164c1db4cbc5d281d00ffa81724a83b3
Final nonce: 1558215
Final hash: 000008fb67e78dee225c2bea554b989b164c1db4cbc5d281d00ffa81724a83b3
```