**AIM**: Implementation of SHA 512 Algorithm.

**PROGRAM CODE:**

import hashlib

# initializing string

str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()

# then sending to SHA256()

result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA256 is : ")

print(result.hexdigest())

print ("\r")

# initializing string

str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()

# then sending to SHA384()

result = hashlib.sha384(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA384 is : ")

print(result.hexdigest())

print ("\r")

# initializing string

str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()

# then sending to SHA224()

result = hashlib.sha224(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA224 is : ")

print(result.hexdigest())

print ("\r")

# initializing string

str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()

# then sending to SHA512()

result = hashlib.sha512(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA512 is : ")

print(result.hexdigest())

print ("\r")

# initializing string

str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()

# then sending to SHA1()

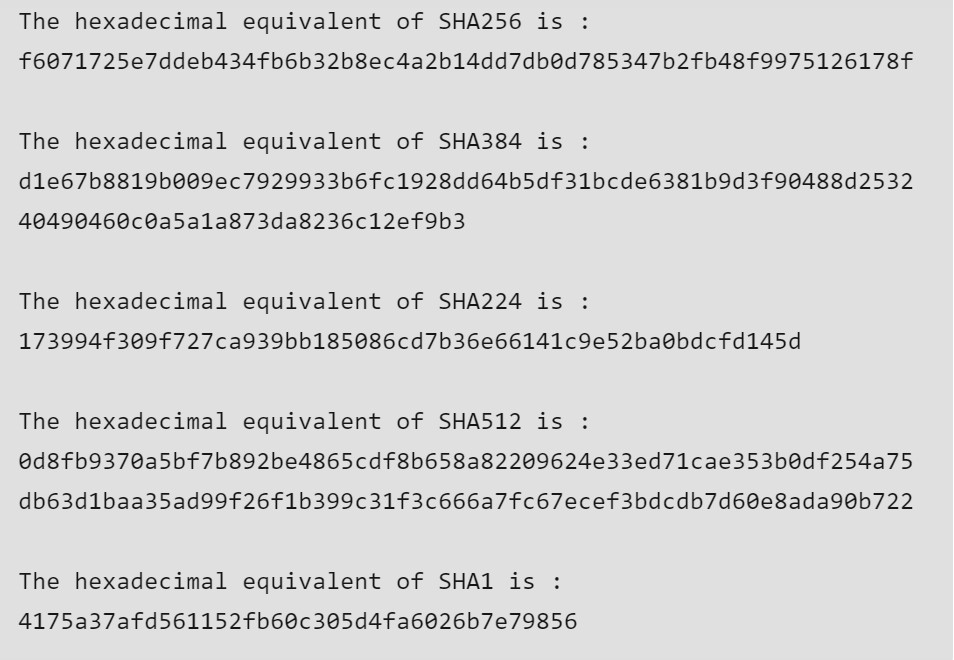
result = hashlib.sha1(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA1 is : ")

print(result.hexdigest())

**OUTCOME:**

****

**Alternate Mehtod:**

Here’s a step-by-step guide to implement the SHA-512 algorithm in Python, based on the details you've provided for Experiment No. 8:

Step 1: Set Up Your Environment

1. Install Python: Ensure you have Python installed on your machine. Use version 3.12 or above for compatibility with the hashlib library.

2. Open Python IDLE: Launch Python IDLE or any preferred code editor.

Step 2: Write the Code

1. Import the hashlib Library: This library provides the SHA-512 and other hashing algorithms.

```python

import hashlib

```

2. Initialize the Input String: Define the string you want to hash.

```python

input\_string = "GeeksforGeeks"

```

3. Create the SHA-512 Hash:

- Encode the string.

- Use `hashlib.sha512()` to generate the hash.

```python

# Encoding input\_string using encode() and sending to SHA512()

sha512\_result = hashlib.sha512(input\_string.encode())

```

4. Print the Hexadecimal Output:

- Use the `hexdigest()` method to get the hash in hexadecimal format.

```python

# Printing the equivalent hexadecimal value for SHA-512

print("The hexadecimal equivalent of SHA512 is:")

print(sha512\_result.hexdigest())

```

Step 3: Run the Code

1. Combine All Steps: Here’s the complete code you will execute in Python IDLE:

```python

import hashlib

# Initializing string

input\_string = "GeeksforGeeks"

# Encoding input\_string using encode() and sending to SHA512()

sha512\_result = hashlib.sha512(input\_string.encode())

# Printing the equivalent hexadecimal value for SHA-512

print("The hexadecimal equivalent of SHA512 is:")

print(sha512\_result.hexdigest())

```

2. Execute the Code: Run the complete code. You should see the output displaying the hexadecimal equivalent of the SHA-512 hash for "GeeksforGeeks".

Step 4: Analyze and Discuss Results

- Outcome: Note the hash value generated and discuss its significance. For instance, you can talk about how the SHA-512 hash is unique to the input string and its applications in data integrity and security.

Step 5: Document Your Experiment

- Write your lab report: Include sections for Aim, Objectives, Theory, Code, Outcome, and Conclusion, just as you have formatted above. You can include observations and any difficulties encountered.

Additional Steps (Optional)

- Explore Other Hash Functions: You can implement and compare other hashing algorithms (SHA-256, SHA-384, etc.) in the same manner as SHA-512 to see how they differ in output.

Example for Additional Hash Functions

Here’s how you can add SHA-256 and SHA-384 for comparison:

```python

# SHA-256

sha256\_result = hashlib.sha256(input\_string.encode())

print("The hexadecimal equivalent of SHA256 is:")

print(sha256\_result.hexdigest())

# SHA-384

sha384\_result = hashlib.sha384(input\_string.encode())

print("The hexadecimal equivalent of SHA384 is:")

print(sha384\_result.hexdigest())

```