UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN

DAA LAB - 7

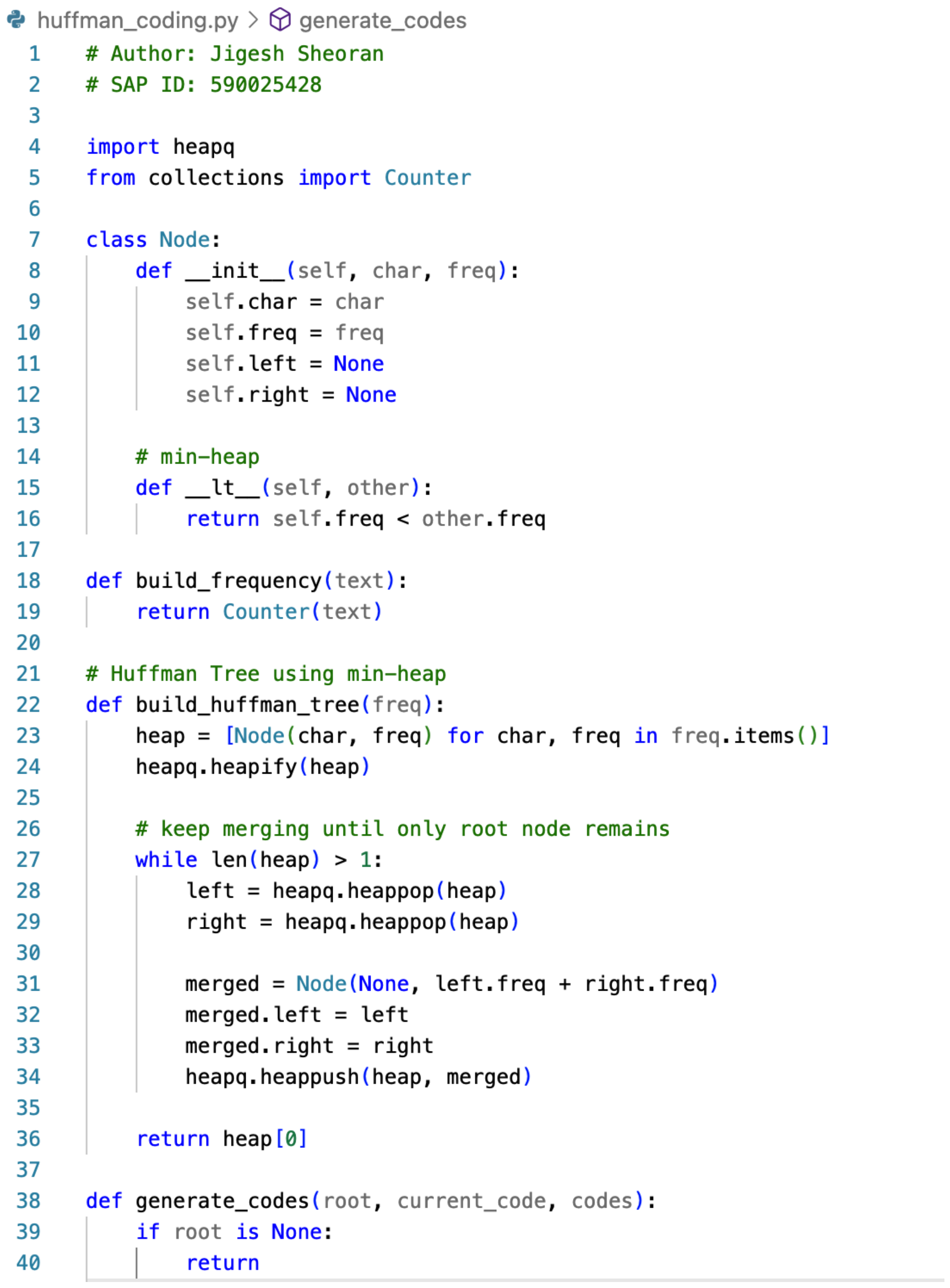
Huffman Coding

MTECH-COMPUTER SCIENCE
ENGINEERING
CYBER SECURITY AND FORENSICS

Name: Jigesh Sheoran
SAP ID: 590025428

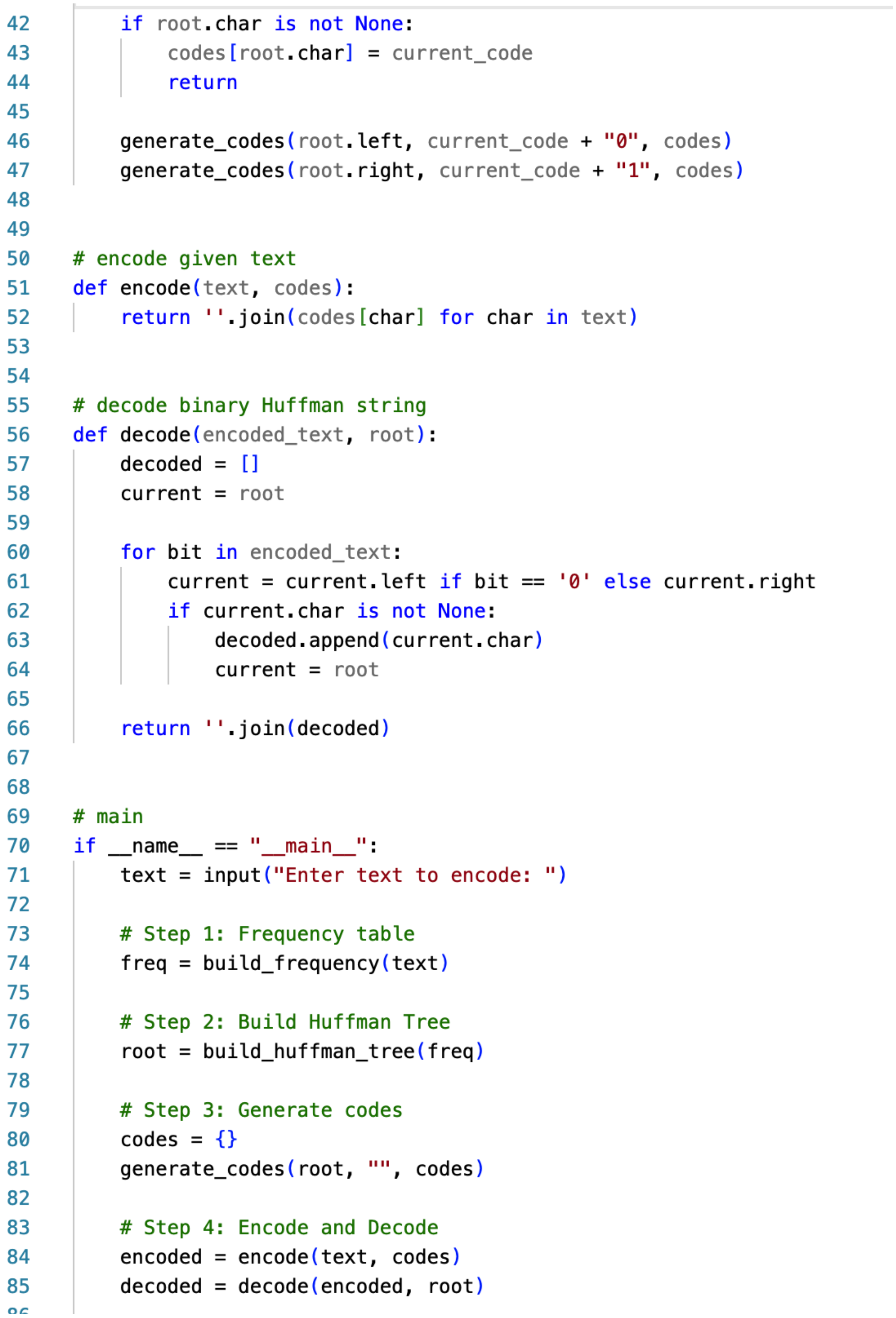
Title: : Implementation of Huffman Coding

Implementation: Language used : Python



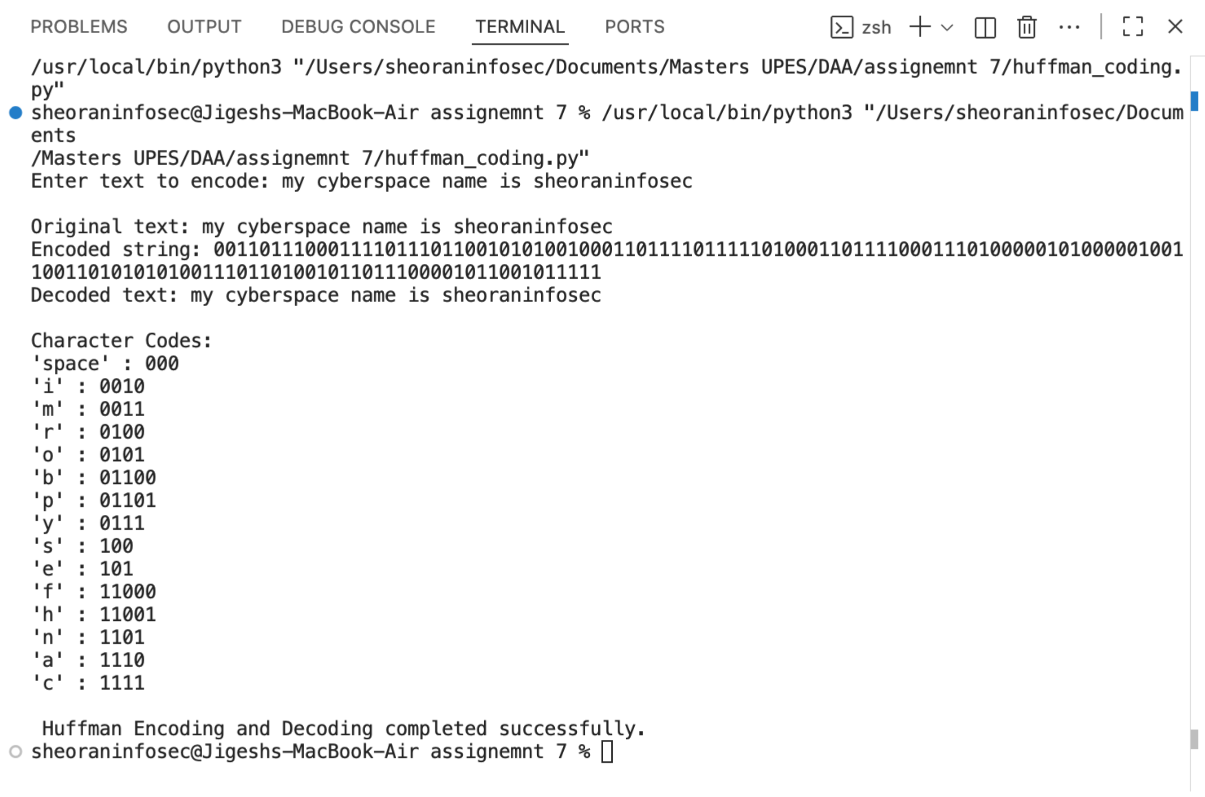
huffman_coding.py > generate_codes

```python
# Author: Jigesh Sheoran
# SAP ID: 590025428

import heapq
from collections import Counter

class Node:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None

    # min-heap
    def __lt__(self, other):
        return self.freq < other.freq

def build_frequency(text):
    return Counter(text)

# Huffman Tree using min-heap
def build_huffman_tree(freq):
    heap = [Node(char, freq) for char, freq in freq.items()]
    heapq.heapify(heap)

    # keep merging until only root node remains
    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)

        merged = Node(None, left.freq + right.freq)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)

    return heap[0]

def generate_codes(root, current_code, codes):
    if root is None:
        return
```



```python
    if root.char is not None:
        codes[root.char] = current_code
        return

    generate_codes(root.left, current_code + "0", codes)
    generate_codes(root.right, current_code + "1", codes)


# encode given text
def encode(text, codes):
    return ''.join(codes[char] for char in text)


# decode binary Huffman string
def decode(encoded_text, root):
    decoded = []
    current = root

    for bit in encoded_text:
        current = current.left if bit == '0' else current.right
        if current.char is not None:
            decoded.append(current.char)
            current = root

    return ''.join(decoded)


# main
if __name__ == "__main__":
    text = input("Enter text to encode: ")

    # Step 1: Frequency table
    freq = build_frequency(text)

    # Step 2: Build Huffman Tree
    root = build_huffman_tree(freq)

    # Step 3: Generate codes
    codes = {}
    generate_codes(root, "", codes)

    # Step 4: Encode and Decode
    encoded = encode(text, codes)
    decoded = decode(encoded, root)

    # Display results
    print("\nOriginal text:", text)
    print("Encoded string:", encoded)
    print("Decoded text:", decoded)
    print("\nCharacter Codes:")
    for char, code in codes.items():
        if char == " ":
            print(f"'space' : {code}")
        else:
            print(f"'{char}' : {code}")

    print("\n Huffman Encoding and Decoding completed successfully.")

```

Output:



```
/usr/local/bin/python3 "/Users/sheoraninfosec/Documents/Masters UPES/DAA/assignemnt 7/huffman_coding.
py"
sheoraninfosec@Jigeshs-MacBook-Air assignemnt 7 % /usr/local/bin/python3 "/Users/sheoraninfosec/Docum
ents
/Masters UPES/DAA/assignemnt 7/huffman_coding.py"
Enter text to encode: my cyberspace name is sheoraninfosec

Original text: my cyberspace name is sheoraninfosec
Encoded string: 0011011100011110111011001010100100011011110111110100011011110001110100000101000001001
1001101010101001110110100101101110000101100101111
Decoded text: my cyberspace name is sheoraninfosec

Character Codes:
'space' : 000
'i' : 0010
'm' : 0011
'r' : 0100
'o' : 0101
'b' : 01100
'p' : 01101
'y' : 0111
's' : 100
'e' : 101
'f' : 11000
'h' : 11001
'n' : 1101
'a' : 1110
'c' : 1111

 Huffman Encoding and Decoding completed successfully.
sheoraninfosec@Jigeshs-MacBook-Air assignemnt 7 %
```

———————————————END OF DOCUMENT ———————————————

JIGESH SHEORAN
590025428
M.Tech CSE