Name: Vinay Hulsurkar
Roll No: 14154

```
import heapq
from collections import defaultdict, Counter

class HuffmanNode:
    def __init__(self, char=None, freq=0):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(self, other):
        return self.freq < other.freq

def build_huffman_tree(freq_map):
    heap = [HuffmanNode(char, freq) for char, freq in freq_map.items()]
    heapq.heapify(heap)

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged = HuffmanNode(freq=left.freq + right.freq)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)
    return heap[0]

def generate_huffman_codes(root):
    codes = {}
    def _generate_codes(node, current_code):
        if not node:
            return
        if node.char is not None:
            codes[node.char] = current_code
            return
        _generate_codes(node.left, current_code + '0')
        _generate_codes(node.right, current_code + '1')
    _generate_codes(root, "")
    return codes

def huffman_encoding(data):
    if not data:
        return "", {}
    freq_map = Counter(data)
    root = build_huffman_tree(freq_map)
    codes = generate_huffman_codes(root)
    encoded_data = ''.join(codes[char] for char in data)

    return encoded_data, codes
```

```
def huffman_decoding(encoded_data, codes):
    if not encoded_data or not codes:
        return ""

    reverse_codes = {code: char for char, code in codes.items()}

    decoded_output = ""
    current_code = ""
    for bit in encoded_data:
        current_code += bit
        if current_code in reverse_codes:
            decoded_output += reverse_codes[current_code]
            current_code = ""
    return decoded_output

if __name__ == "__main__":
    data = "Vinay Hulsurkar"
    print(f"Original Data: {data}")

    encoded_data, codes = huffman_encoding(data)
    print(f"\nHuffman Codes: {codes}")
    print(f"\nEncoded Data: {encoded_data}")

    decoded_data = huffman_decoding(encoded_data, codes)
    print(f"\nDecoded Data: {decoded_data}")
```

Output:

Original Data: Vinay Hulsurkar

Huffman Codes: {'l': '000', 'r': '001', 's': '0100', 'y': '0101', 'a': '011', 'V': '1000', ' ': '1001', 'H': '1010', 'n': '1011', 'k': '1100', 'i': '1101', 'u': '111'}

Encoded Data: 10001101101101101011001101011100001001110011100011001

Decoded Data: Vinay Hulsurkar