```
import heapq

class HuffmanNode:
    def __init__(self, cha, F
        self.cha = cha
        self.Frequency = Freq
        self.left = None
        self.right = None

    def __lt__(self, other):
        if self.Frequency ==

            if self.cha is No
                return False
            if other.cha is N
                return True
            return self.cha.
        return self.Frequency

# Using MinHeap as the primar
class MinHeap:
    def __init__(self):
        self.heaps = []

    def push(self, node):
        heapq.heappush(self.h

    def pop(self):
        return heapq.heappop(

    def size(self):
        return len(self.heaps

class HuffmanCodingTree:
    def __init__(self):
        self.root = None
        self.codes = {}
        self.mini_heap = MinH
        self.mini_heap = Min

    def build_tree(self, cha_
        # Now,push all charac
        for cha, Frequency in
            self.mini_heap.pu

        # Building a tree usi
        while self.mini_heap.
            left = self.mini_
            right = self.mini

            internal = Huffma
            internal.left = l
            internal.right =

            self.mini_heap.pu

        self.root = self.mini

    def generate_codes(self):
        def dfs(node, current
            if node is None:
                return

            if node.cha is no
                self.codes[no
                return

            dfs(node.left, cu
            dfs(node.right, c

        dfs(self.root, "")

    def print_tree(self, node
        if node is None:
            node = self.root

        if node.cha is not No
            print(f"{
```

```
            print(f"{prefix}
        else:
            print(f"{prefix}
            self.print_tree(r
            self.print_tree(r

def get_user_input():
    cha_Frequency = {}
    print("Enter character-fr
    while True:
        cha = input(" Enter (
        if not cha:
            break
        try:
            Frequency = int(i
            if Frequency <= 0
                print("Freque
                continue
            cha_Frequency[cha
        except ValueError:
            print("Invalid fr
    return cha_Frequency

def main():
    cha_Frequency = get_user_

    if not cha_Frequency:
        print("Input was not
        return

    huffman_tree = HuffmanCod
    huffman_tree.build_tree(c
    huffman_tree.generate_cod


    print("\nHuffman Tree:")
    huffman_tree.print_tree()

    print("\nHuffman Codes:")
    for cha, code in sorted(
        print(f"{cha}: {code

if __name__ == "__ma
    main()
```

```
Enter character-frequency pairs (press Enter without input to finish):
 Enter Character or press enter if completed:
Input was not provided.So,Program is exited.
```

Double-click (or enter) to edit