import java.util.PriorityQueue;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Node implements Comparable<Node> {

char data;

int frequency;

Node left;

Node right;

public Node(char data, int frequency) {

this.data = data;

this.frequency = frequency;

}

@Override

public int compareTo(Node other) {

return Integer.compare(this.frequency, other.frequency);

}

}

class HuffmanCoding {

static Map<Character, String> huffmanCodes = new HashMap<>();

public static void generateHuffmanCodes(Node root, String code) {

if (root == null) {

return;

}

if (root.left == null && root.right == null) {

huffmanCodes.put(root.data, code);

}

generateHuffmanCodes(root.left, code + "0");

generateHuffmanCodes(root.right, code + "1");

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String s = scanner.nextLine();

Map<Character, Integer> freq = new HashMap<>();

for (char c : s.toCharArray()) {

freq.put(c, freq.getOrDefault(c, 0) + 1);

}

PriorityQueue<Node> pq = new PriorityQueue<>();

for (Map.Entry<Character, Integer> entry : freq.entrySet()) {

pq.add(new Node(entry.getKey(), entry.getValue()));

}

while (pq.size() > 1) {

Node l = pq.poll();

Node r = pq.poll();

Node n = new Node('\0', l.frequency + r.frequency);

n.left = l;

n.right = r;

pq.add(n);

}

generateHuffmanCodes(pq.poll(), "");

System.out.println("Huffman Codes:");

for (Map.Entry<Character, String> entry : huffmanCodes.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

}

}