LP3 (DAA) Lab Exp No.2

import java.util.\*;

class Node implements Comparable<Node> {

    int val;

    String symbol;

    String huffman;

    Node left;

    Node right;

    public Node(int val, String symbol) {

        this.val = val;

        this.symbol = symbol;

        this.huffman = "";

        this.left = null;

        this.right = null;

    }

    public int compareTo(Node other) {

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

    }

}

public class Huffman {

    public static Node buildHuffmanTree(String[] characters, int[] frequencies) {

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

        for (int i = 0; i < characters.length; i++) {

            nodes.offer(new Node(frequencies[i], characters[i]));

        }

        while (nodes.size() > 1) {

            Node left = nodes.poll();

            Node right = nodes.poll();

            left.huffman = "0";

            right.huffman = "1";

            Node newNode = new Node(left.val + right.val, left.symbol + right.symbol);

            newNode.left = left;

            newNode.right = right;

            nodes.offer(newNode);

        }

        return nodes.poll();

    }

    public static void printHuffmanCodes(Node node) {

        HashMap<String, Integer> huffmanDecimal = new HashMap<>();

        traverse(node, "", huffmanDecimal);

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

        for (Map.Entry<String, Integer> entry : huffmanDecimal.entrySet()) {

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

        }

    }

    private static void traverse(Node node, String currentCode, HashMap<String, Integer> huffmanDecimal) {

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

            huffmanDecimal.put(node.symbol, Integer.parseInt(currentCode, 2));

        } else {

            if (node.left != null) {

                traverse(node.left, currentCode + node.left.huffman, huffmanDecimal);

            }

            if (node.right != null) {

                traverse(node.right, currentCode + node.right.huffman, huffmanDecimal);

            }

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter characters (separated by space): ");

        String inputChars = scanner.nextLine();

        String[] characters = inputChars.split(" ");

        System.out.print("Enter corresponding frequencies (separated by space): ");

        String inputFreqs = scanner.nextLine();

        String[] freqStrings = inputFreqs.split(" ");

        int[] frequencies = new int[freqStrings.length];

        for (int i = 0; i < freqStrings.length; i++) {

            frequencies[i] = Integer.parseInt(freqStrings[i]);

        }

        Node root = buildHuffmanTree(characters, frequencies);

        printHuffmanCodes(root);

        scanner.close();

    }

}

OUTPUT: -

