ASSIGNMENT NO. : 2

import java.util.PriorityQueue;

class Node {

int freq;

String symbol;

Node left, right;

String huff = "";

public Node(int freq, String symbol) {

this.freq = freq;

this.symbol = symbol;

}

// Comparator for priority queue

public static class NodeComparator implements java.util.Comparator<Node> {

public int compare(Node x, Node y) {

return x.freq - y.freq;

}

} }

public class HuffmanCoding {

// Function to print Huffman codes

public static void printNodes(Node root, String val) {

String newVal = val + root.huff;

// If this is a leaf node

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

System.out.println(root.symbol + " -> " + newVal);

}

if (root.left != null) {

root.left.huff = "0";

printNodes(root.left, newVal);

}

if (root.right != null) {

root.right.huff = "1";

printNodes(root.right, newVal);

}

}

public static void main(String[] args) {

String[] chars = {"a", "b", "c", "d", "e", "f"};

int[] freqs = {5, 9, 12, 13, 16, 45};

PriorityQueue<Node> queue = new PriorityQueue<>(new Node.NodeComparator());

// Create initial nodes

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

queue.add(new Node(freqs[i], chars[i]));

}

// Build the Huffman tree

while (queue.size() > 1) {

Node left = queue.poll();

Node right = queue.poll();

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

queue.add(newNode);

left.huff = "0";

right.huff = "1";

newNode.left = left;

newNode.right = right;}

printNodes(queue.poll(), "");

}}

**OUTPUT**: f -> 0

c -> 100

d -> 101

a -> 1100

b -> 1101

e -> 111