import heapq

from collections import defaultdict

class HuffmanNode:

def \_\_init\_\_(self, freq, symbol, left=None, right=None):

self.freq = freq

self.symbol = symbol

self.left = left

self.right = right

def \_\_lt\_\_(self, other):

return self.freq < other.freq

def huffman\_encoding(symbols, frequencies):

heap = [HuffmanNode(frequencies[i], symbols[i]) for i in range(len(symbols))]

heapq.heapify(heap)

while len(heap) > 1:

left = heapq.heappop(heap)

right = heapq.heappop(heap)

merged = HuffmanNode(left.freq + right.freq, left.symbol + right.symbol, left, right)

heapq.heappush(heap, merged)

root = heap[0]

codes = {}

generate\_codes(root, "", codes)

return codes

def generate\_codes(node, current\_code, codes):

if node is None:

return

if not node.left and not node.right:

codes[node.symbol] = current\_code

generate\_codes(node.left, current\_code + "0", codes)

generate\_codes(node.right, current\_code + "1", codes)

n = int(input("Enter number of symbols: "))

symbols = []

frequencies = []

for i in range(n):

symbol = input(f"Enter symbol {i+1}: ")

frequency = int(input(f"Enter frequency of symbol {symbol}: "))

symbols.append(symbol)

frequencies.append(frequency)

codes = huffman\_encoding(symbols, frequencies)

print("Huffman Codes:", codes)