Project 3: Tree: Huffman Tree

Problem Description

Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. Generating huffman codes for a set of symbols by the binary tree can be formalized as follows[1]:

- **Input:**
  Alphabet $A = \{a_1, a_2, \cdots, a_n\}$, which is the symbol alphabet of size n.
  Tuple $W = \{w_1, w_2, \cdots, w_n\}$, which is the tuple of the (positive) symbol weights (usually proportional to probability/frequency), i.e. $w_i = \text{weight}(a_i), 1 \leq i \leq n$.
- **Output:**
  Code $C(W) = \{c_1, c_2, \cdots, c_n\}$, which is the tuple of (binary) codewords, where c_i is the codeword for $a_i, 1 \leq i \leq n$.
- **Goal:**
  Let $L(C(W)) = \sum_{i=1}^{n} w_i \times \text{length}(c_i)$ be the weighted path length of code C. Condition: $L(C(W)) \leq L(T(W))$ for any code $T(W)$.

Now, the table below provides the characters and their frequencies in the movie quote "*choose me let me make you happy*". You need to write a huffman coding program which supports to encode this quote into a bitstream and decode a given bitstream to the string :-)

Table 1: Characters and their frequencies.

c	s	l	t	k	u	h	a	y	p	o	m	e	-
1	1	1	1	1	1	2	2	2	2	3	3	5	6

All the following functions are required in your program:

(1) HuffmanCode: Build a huffman tree and print the huffman code of each character.

(2) Encoder: Take the quote as input and output the bitstream.

(3) Decoder: Take bitstream as input and output the original text.

Note: You may need a priority queue and design a huffman tree node for the implementation.

[1] https://en.wikipedia.org/wiki/Huffman_coding