Assignment 4

1. **Explain the implementation of the Huffman Tree**

To put it briefly, Huffman coding is essentially calculating the frequency of each letter as an input text, which means that each character has a value. Each character is represented as a leaf node according to its frequency. All of these nodes are presented as priority queues (min-heap), with the nodes arranged lexicographically by their characters as well as according to their frequency and equal frequencies. Additionally, this tree is continuously replicated depending on two nodes, with the little frequencies merging into a new internal node and essentially being pulled back into the merged node until just the root is left. Once this tree has been built, the program will recursively traverse to assign binary Huffman codes to each character. This means that the left branches are represented by zeros, and the right branches by ones.

2. **Explain the results of the Huffman Code**

The frequency of each character and its matching Huffman code are displayed in a table for our findings. These particular codes essentially encrypt any text or user input we add to our program, compressing it into a binary string that is then decoded. Additionally, this application determines the original text size in bits as well as the compressed size, displaying a compression ratio that illustrates the actual efficiency of human coding. Generally speaking, the following findings show how often occurring letters enable shorter codes, while less frequently occurring characters do the opposite, resulting in efficient compression with any input text that we supply to the software.

Example input texts from the program:

```
PS D:\Code\Assignment 4>  d:; cd 'd:
6eb04089b24446ed9afeaff225\redhat.ja
Enter text to encode:
abracadabra
Text: ABRACADABRA

Output:

Character  Frequency  Huffman Code
A          5          0
B          2          110
R          2          111
C          1          100
D          1          101

Encoded: 01101111010001010111011110
Original Bits: 11 * 8 = 88 bits
Compressed Bits: 23 bits
Compression Ratio: 3.83 : 1
Decoded Text: ABRACADABRA
PS D:\Code\Assignment 4> ^C
PS D:\Code\Assignment 4>
PS D:\Code\Assignment 4>  d:; cd 'd:
6eb04089b24446ed9afeaff225\redhat.ja
Enter text to encode:
conrad
Text: CONRAD

Output:

Character  Frequency  Huffman Code
A          1          00
C          1          100
D          1          101
N          1          01
O          1          110
R          1          111

Encoded: 1001100111100101
Original Bits: 6 * 8 = 48 bits
Compressed Bits: 16 bits
Compression Ratio: 3.00 : 1
Decoded Text: CONRAD
```