

REPORT – PROGRAM ASSIGNMENT 2

1. Description:

Huffman Coding is a commonly used algorithm for lossless data compression.

Characters are compressed based on their occurrence frequency.

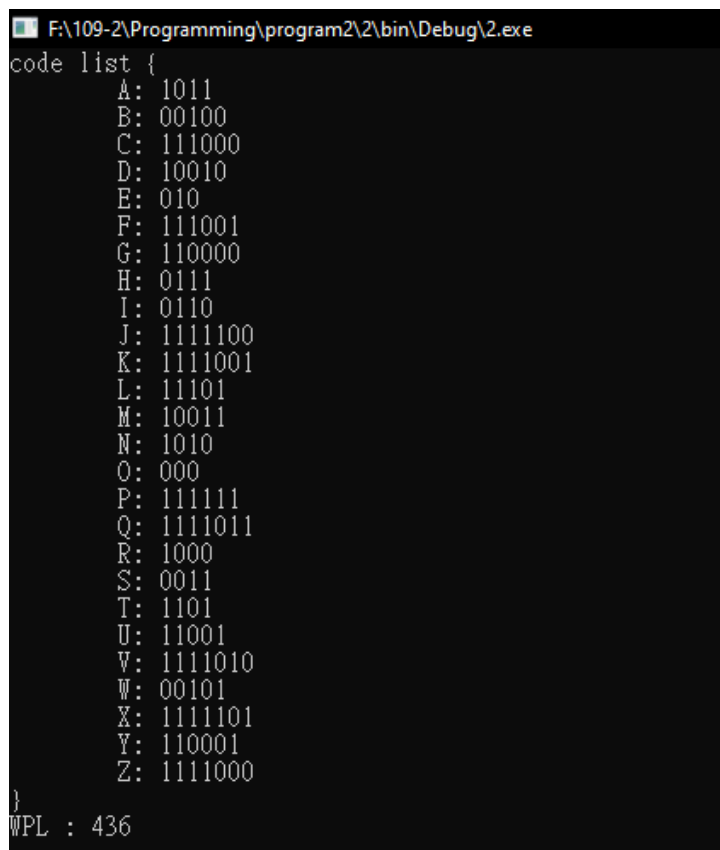
Characters that occur more often are represented using fewer bits than characters that appear less frequently.

Create a priority queue containing the characters and the frequency of occurrence (according to the criteria of smallest occurrence).

Create a new node, the smallest value is in the left Child node, the second smallest value is in the right Child node. The sum of the two smallest values will be the value of the new node. Insert this New node into the tree.

From the root Node, we can find the encoded string of the character.

2. Part 1 Output:



```
F:\109-2\Programming\program2\2\bin\Debug\2.exe
code list {
  A: 1011
  B: 00100
  C: 111000
  D: 10010
  E: 010
  F: 111001
  G: 110000
  H: 0111
  I: 0110
  J: 1111100
  K: 1111001
  L: 11101
  M: 10011
  N: 1010
  O: 000
  P: 111111
  Q: 1111011
  R: 1000
  S: 0011
  T: 1101
  U: 11001
  V: 1111010
  W: 00101
  X: 1111101
  Y: 110001
  Z: 1111000
}
WPL : 436
```

3. Part 2 Output:

Input: iaaaaaaamhhhhannnnddsssomeeeee

```
F:\109-2\Programming\program2\2.2\bin\Debug\2.exe
Enter character: iaaaaaaamhhhhannnnddsssomeeeee
encoding result: 00001010101010101010100011011011011100100100100101011011010010010010001110011111111111111
code list{
    a: 10
    d: 1101
    e: 111
    h: 011
    i: 0000
    m: 1100
    n: 010
    o: 0001
    s: 001
}
WPL = 88
decoding result: iaaaaaaamhhhhannnnddsssomeeeee
```

Input: howwwarreyooou

```
F:\109-2\Programming\program2\2.2\bin\Debug\2.exe
Enter character: howwwarreyooou
encoding result: 011110000000011011111111111010101010101100
code list{
    a: 0110
    e: 1101
    h: 0111
    o: 10
    r: 111
    u: 1100
    w: 00
    y: 010
}
WPL = 42
decoding result: howwwarreyooou
```

4. Explanation:

In part 1, I created 2 arrays containing the characters and the frequency of occurrence. The node will be created with a pointer to the character and frequency values, leftChild and rightChild. Then use the Huffman Coding algorithm to generate a binary tree. To find the encoding, from the root Node to find the character (not the "*"), if to the leftChild the encoding will be +"0", rightChild will be +"1". (Use recursion). The encoded segment will be put into an array of values (A-Z). Finally print the array of values (guaranteed from A-Z).

In part 2, I use map to store values (input and code list). The way to get the code list as part 1. Encoding result, run the string (input) and the result will be += the encoding value of at the character being considered. Encoding result, keeping a temporary value as the root, From the encoding string, "0" moves to the left child, "1" moves to the rightChild until leftChild and rightChild are null. The result will be += the character of the current node. Update the temporary value as root. Do it until the end.