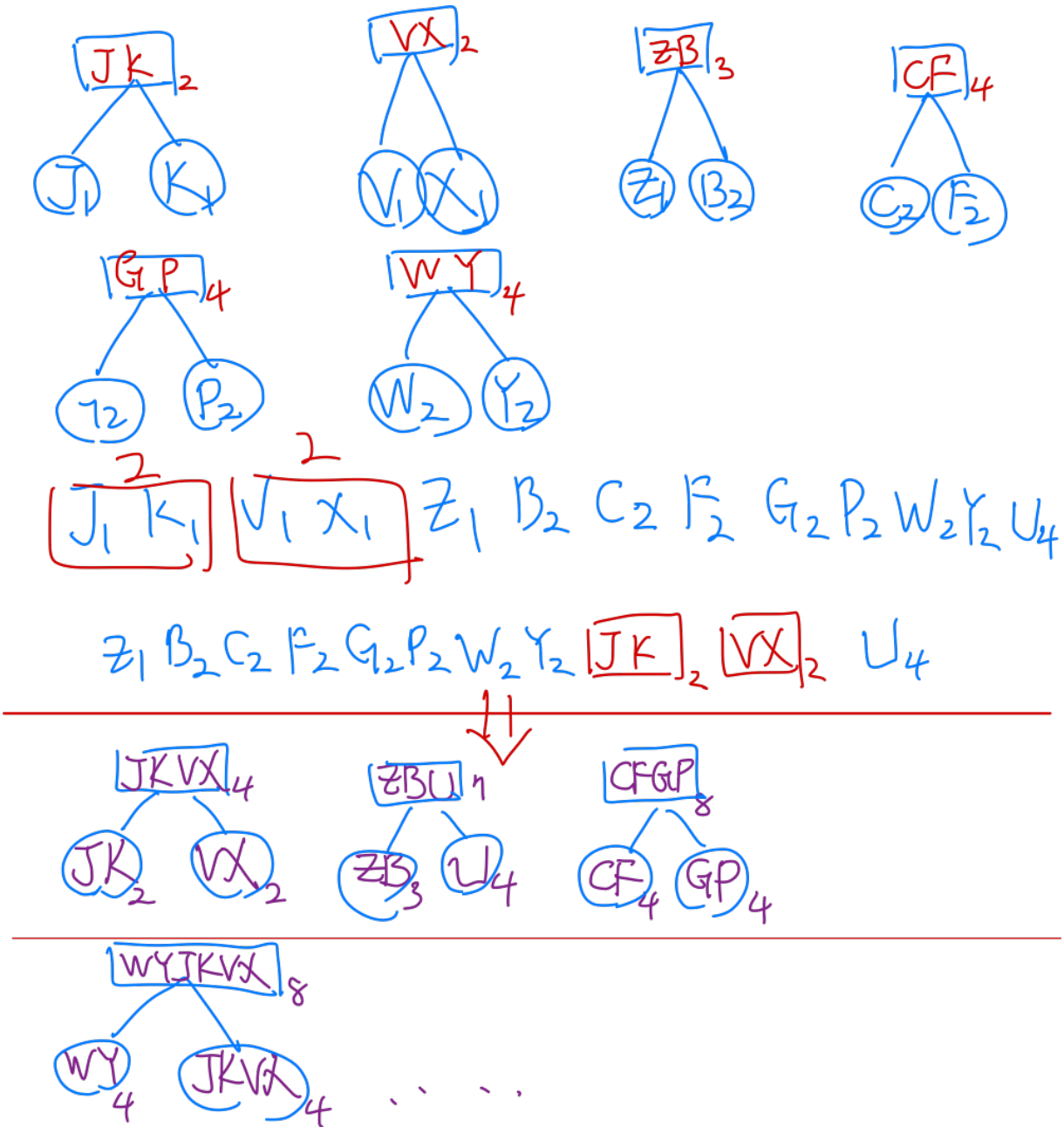
Program Assignment 2

StudentID: 0816067 Name: 李昀澤

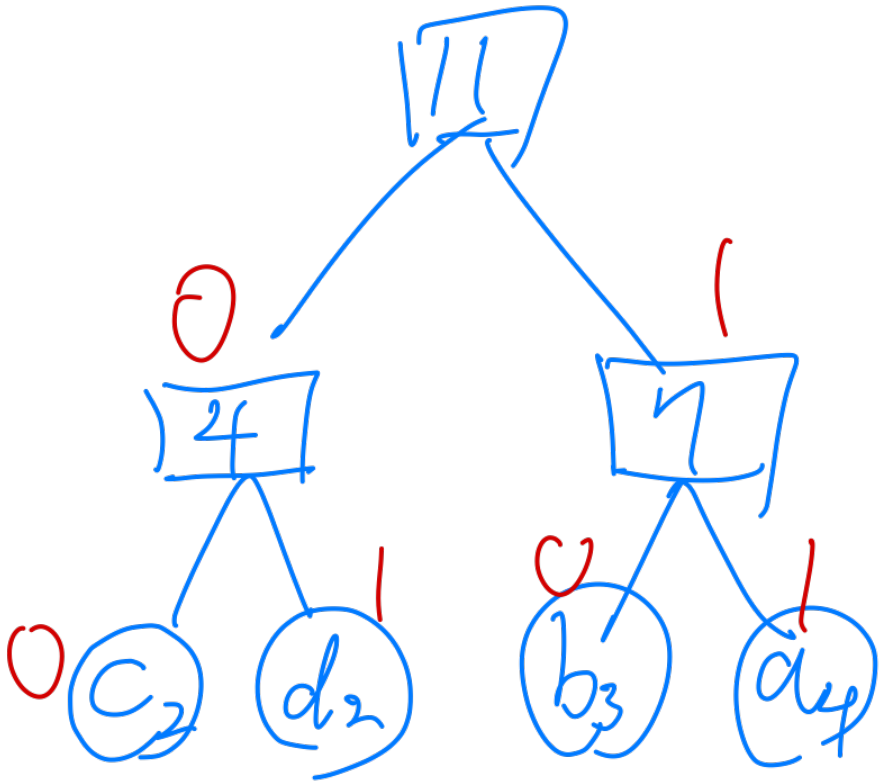
1. How to do Huffman encoding?

This is a data-compressing skill with no data distortion. With this manner, we can compress a text file efficiently and economically. To implement this encoding, the first thing to do is construct a Huffman tree. The detail of the construction has been included in the **Huffman Coding.pptx**, so here I just construct some pieces by handwriting below:



In a word, I first the times of occurrence of an alphabet, then put those information into a data structure called forest. To deal with it, I pick two alphabets with the least frequency of occurrence (This step has been completed in the question of Part.1, so we just need to implement the following steps), and make a **parent node** with the frequency from the addition of their frequency, then put this parent node into the forest and pop the two child nodes from the forest. Repeat these steps until there is only one tree in the forest which is the **root**, and it is the tree we call **Huffman Tree**. Once the forest is completed, the alphabets will only appear on the leaf nodes, so we can make its huffman code from the Huffman Tree like below:

a = 4 b = 3 , c = 2 , d = 2



a: 11 , b: 10 , c: 00. d: 01

After constructing the Huffman Tree and the Huffman Code, the encoding will be very simple. We just read the alphabets from input and output the Huffman Code of the alphabets from input. The decoding is the reverse. My way to do decoding is read the encoded code and then compare with the Huffman code from the Huffman Tree. Once finding a totally same string between encoded code and Huffman code, put the alphabet with the certain Huffman code into the decoding output. This is my way to do the Huffman encoding and decoding.

2. The result of Part.1

```
code list{
    A:0101
    B:000111
    C:000110
    D:01101
    E:101
    F:000101
    G:000100
    H:1100
    I:1001
    J:0000001
    K:0000000
    L:00101
    M:01100
    N:0100
    O:111
    P:000011
    Q:110111
    R:1000
    S:0111
    T:0011
    U:00100
    V:110110
    W:000010
    X:110101
    Y:000001
    Z:110100
}
WPL: 436
[Finished in 1.2s]
```

3. The result of Part.2

string: iaaaaaaamhhhhannnnddsssomeeeee

```
Enter character:
Encoding result: 0001010101010101011101100100100100011011011011011100110000100100100001101111111111111111
code list{
    a:01
    d:1100
    e:111
    h:100
    i:0001
    m:1101
    n:101
    o:0000
    s:001
}
WPL: 88
Decoding_result: iaaaaaaamhhhhannnnddsssomeeeee
[Finished in 1.1s]
```

string: howwwarrreyooou

```
Enter character:
Encoding result: 0101011111111111010000011000111101010011 0
code list{
    a:1101
    e:1100
    h:010
    o:10
    r:00
    u:0110
    w:111
    y:0111
}
WPL: 42
Decoding_result: howwwarrreyooou
[Finished in 0.8s]
```