For input files checkpoint1, the following is the encoded output :
Header Begin:
10 1
97 3
File Begin:
0001

Construct a Huffman coding tree:

```
        [S: /n, C:4]
        /          \
[S: a, C:3]     [S: /n, C:1]
```

a:0, /n:1
aaa/n : 0001

For input files checkpoint2, the following is the encoded output :
Header Begin:
10 1
101 1
104 1
108 2
111 1
File Begin:
01011000001110

Construct a Huffman coding tree:

```
                                 [ S: e    ,      C: 6]
                               /                       \
                             /                            \
                [S: o, C: 4]                              [S: e, C: 2]
                 /          \                             /           \
              /          [S: o, C: 2]             [S: /n, C: 1]   [S: e, C: 1]
           /             /           \
[S: l, C: 2]    [S: h, C: 1]     [S: o, C: 1]
```

l:00, h:010, o:011, /n:01, e:11
hello/n : 01011000001110

Build the tree: Create HCNode by the vector of frequencies —> push all the HCNodes into priority queue —> pop first two high priority HCNodes to combine a new HCNode with the sum of their counts and symbol of higher priority one —> if there are still some HCNodes in priority queue, push this new HCNode into the priority queue and continue popping first two high priority HCNodes. Else there are no HCNode in priority queue, let root point to the new node.

Find the code word: Get the symbol and find the pointer to that leaf HCNode —> Traverse the parent and record “1” or “0” until reaching the root and output the reverse version of record that is the code of this word. —> If there are still some bytes, get next symbol.

Lijun Chen
A53071897