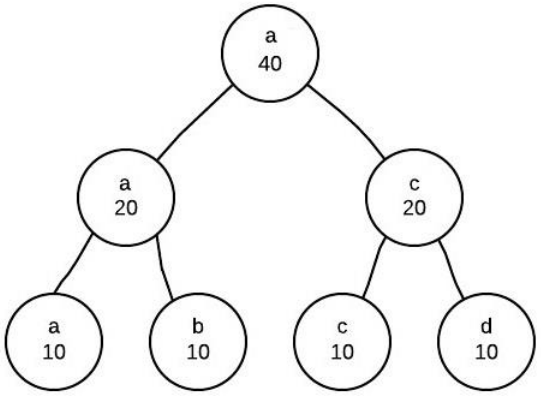
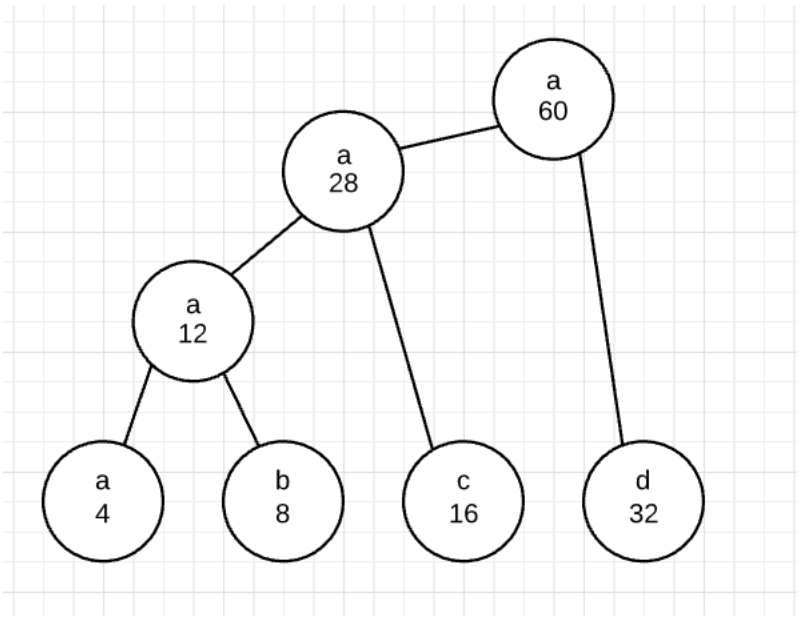
Checkpoint1

Line number	Output
1~97	0
98	10
99	10
100	10
101	10
102~256	0
257	00011011000110110001101100011011000110110001101100011011000110110001101100011011 00011011



Checkpoint2

Line number	Output
1~97	0
98	4
99	8
100	16
101	32
102~256	0
257	00000000100100100101010101010101011111111111111111111111111111111101010101 01010101001001001001000000



How you build the tree and How you find the code word for each byte

Before build the tree, I save the frequency counts of the bytes in the original uncompressed file as a sequence of 256 integers at the beginning of the compressed file (creating a header to the output file to enable the coding tree to be reconstructed when the file is read by the uncompress program). After successfully create the header into the output file, I call a build function which will firstly create a priority queue and create nodes with non-zero frequency counts. After that, store nodes into leaves vector(already created) and push those leaves into the priority queue.

Once priority queue is filled with non-zero frequency leaves' nodes, while size of the priority is not zero, make child0 node pointer to point higher priority node in the priority queue and pop it. Make child1 node pointer to point next higher priority node the priority queue and pop it. Make a parent node that has sum of child0 node and child1 frequencies and symbol of child0 to make break tie. Make the parent node points to child0 and child1 and make child nodes points to the parent node. Push the parent node into the priority queue. Repeat following steps until the size of the priority queue becomes one and make root pointer to the last node remained in priority queue.

Read one byte from the file and store as unsigned char by calling get() function(from ifstream). Call function encode() with stored unsigned char values. Make a node pointer curr points to a leave node depends on symbol. If curr is equal to child0, put 0 into stack. If curr is equal to child1, put 1 into stack. Once curr is equal to root, stop the while loop. After that, write stored values in stack to out file.

To find the code word for each byte, reconstruct the tree with the header and then use decode() function. In decode() function, make a node pointer curr points to root. While curr has child0 or has child1, store in.get() value to a int variable. If value of the variable is 0, move curr pointer to child0. I fvlaue of the variable is 1, move curr pointer to child1. If curr does not have child0 or child1, stop the loop and return symbol of current node pointing.