First run the input file : checkpoint1.txt, The following is the encoded output:
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

10
10
10
10
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

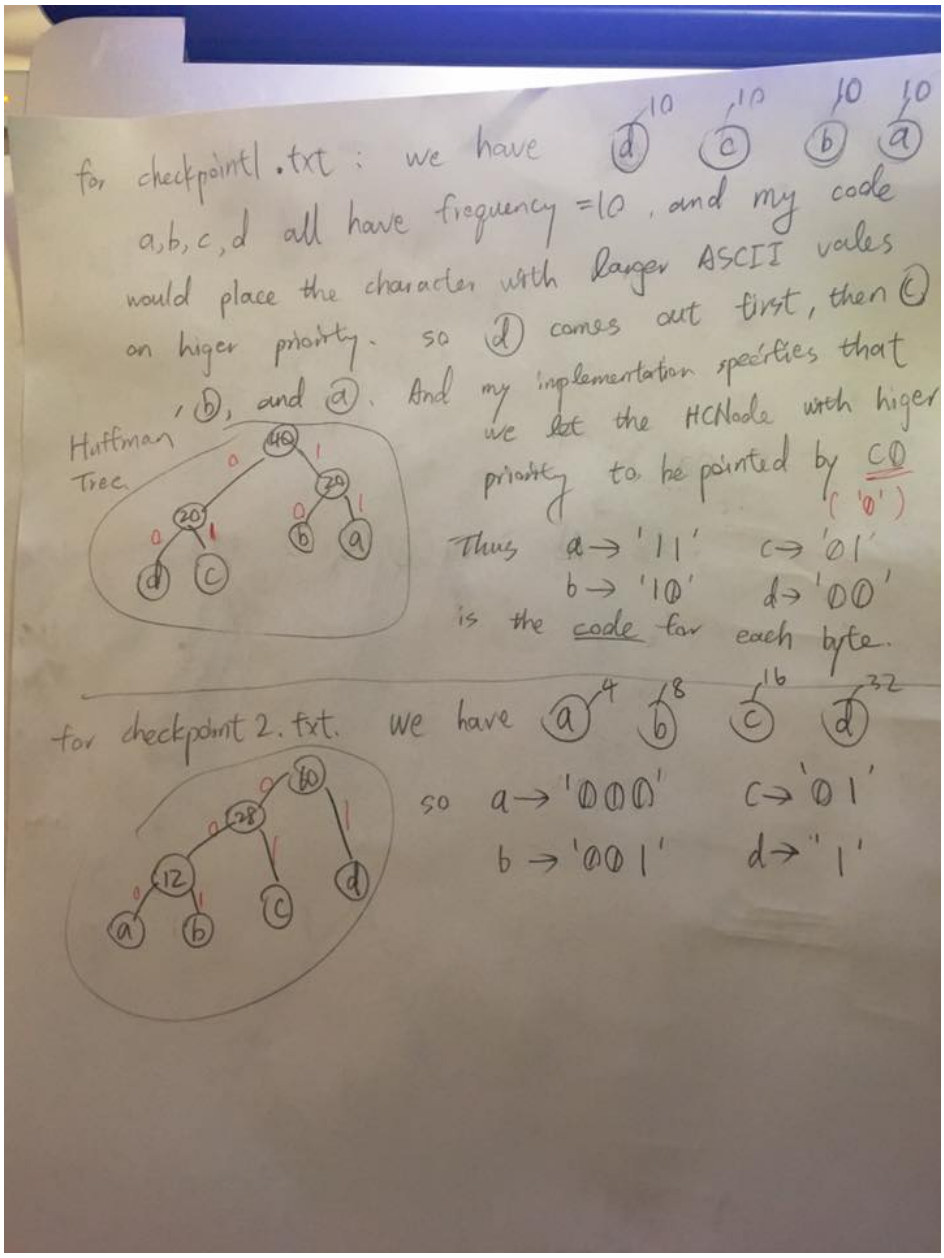
0
0
0
0
0
0
0
0
0
0
0
0
1110010011100100111001001110010011100100111001001110010011100100111001
00

Next, for checkpoint2.txt

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
4
8
16
32
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
00000000100100100101010101010101011111111111111111111111111111111111110101010101010
1010010010010010000000



Now, since I've already written out the code for each symbol, we can easily compare them with the encoded output(which is the row of 0's and 1's immediately after the header). Luckily, I got my compress.cpp and uncompress.cpp working the first time after I finished implementing them. And I've also tested for the edge case such as input file being empty(my program would print out an error message and return 0 in this case) and the file containing single character(possibly repeated many times).

The umcompress.cpp has also been tested and it does return completely identical message to the original input file before being encoded.

It's worth mentioning that the biggest obstacle/bug I encountered for the checkpoint was actually about the std::priority_queue. I did not initialize them in the compress.cpp file and instead I put it as a member variable of HCTree class. It took me a great deal of time until I finally got it working properly.