Set Spec decisions

Use this section to keep track of set-level decisions on the spec.
Original spec on D2L

INNER WORKINGS OF HUFFMAN
ENCODING
1- count the number of appearances of every character(uint8_t) in the file
2- create a binary tree node for every character, the node should contain a variable to store the character and the count of the character. This should be stored in an array
3- sort the node array
4- pop the two nodes(B) with the smallest value. Create a new node(R), to the character assign **anything**, assign each of the popped nodes(B) to each of the branches, assign the sum of the counts in both popped nodes(B) as the count of the new node(R)
5- insert the new node(R) back into the array
6- repeat 3 through 5 until there is only one root node in the array and all of the initial nodes are now leaves(HUFFMAN TREE)
7-WRITE the huffman tree as specified in ISSUE 3
7- read a character
8- evaluate the path from the root of the tree up to the leaf where the character is.
10- write the path from the root to the character to the file. For every time the left branch is chosen, write a 0 bit, for the right branch write a 1 bit
Note: You will need to hold on to your bits until you have at least 8 before outputting.
11- DO NOT WRITE THE CHARACTER
12- do 7 through 11 until the input file has been completely read
13- you are done

DECODING
1-read the first character, store for future comparison
2- read the appearance count
3- make a tree node with the character and the appearance(see encoding step 2)
4- read the next character. If it is the same as the first character then the header is over GOTO step 8, else continue
5- read the appearance count of the character
6- create node in analogue manner to ENCODE step 2 and insert into array
7- GOTO step 4
8- you now have all the character nodes inside array. Create huffman tree in analogous manner to ENCODING
10 - evaluate the number of relevant bit according to calculation 1.
9- read “bit” from file (remember a bit is actually a ‘0’ or ‘1’ char) if there are no more bits to read you are done
10 - if 0 move into the left branch of the current root, if 1 move to the right
11 - if the node you moved into is a leaf then return the character contained in the node else GOTO step 9

ISSUE 1
Numerical value of left vs right branch(encoding step 9)
SOLUTION
Left branches are assigned 0, right branches, 1
REASON FOR SELECTION
No specific reason other than the courses I have participated in favor this notation

ISSUE 2
Ordering of bits from root to leaf. Eg root -> 0 -> 1 -> leaf **vs** leaf -> 1 -> 0 -> root
SOLUTION
Encoded paths assume that the root is at the left with every subsequent bit representing the decision to move left or right down the path

```
   Eg 0111
   /\
  0
 /  \
    1
     \
      1
       \
        1
```
REASON FOR SELECTION
This ordering will make implementation easier to decode

ISSUE 3
Encoding of huffman tree
SOLUTION
WRITE each character followed by appearance count at the beginning of the file. The end of the header of the file will be signaled by the repetition of the first character written in the file in the following manner:
(1B)first_character(4B)count of appearances(1B)character(4B)count of appearances…(1B)first_character |
...content…

ISSUE 4
How do you know when you should stop reading?
SOLUTION
The first byte in the file will tell how many relevant bits are contained in the last byte of the file. You should read the file and evaluate from the length of each codeword and the frequency how many total bits you will need.(You should have this after you built the Huffman tree). The total bits can be moduled with 8 and the remainder is number of relevant bits that will spill over to the last byte

REASON FOR SELECTION
- Simplest implementation. The same algorithm used to create the Huffman tree after counting the number of characters in the file can be reused from the information in the file
- Storage cost was not taken into consideration
- 4B was chosen to represent the character count as this would enable the file to contain up to 2^32-1 occurrences of the character which is more than enough
- The character is placed before the count in order to facilitate the identification of the end of the header

ISSUE 5
Endianism. Are you reading each byte from right to left or left to right.
SOLUTION
Read from left to right thus in 11100000 if only 2 bits are relevant, 100000 can be discarded and the two relevant bits are 11

ISSUE 6 (ADDED Sat, oct 24, 2pm) (LAST MODIFIED(SAT oct 24, 2:40pm)
2 characters have the same count of appearances
SOLUTION
Sort the characters by their actual value
E.g.
CHAR 3 COUNT 4
CHAR 9 COUNT 4
3 < 9 so sorted in ascending order: … , 3, 9, …

ISSUE 7 (ADDED Sat, oct 24, 2:20pm)
How to sort two nodes with the same value
SOLUTION
Insert in the smaller value side of the node of equal value that is already present:

```
1          2         3         3          6

                3             3          6
               / \
             1     2

             3         6         6
                      / \
                     3    3
                    / \
                  1     2

              6          9
                        / \
                       3    6
                           / \
                          3    3
                         / \
                        1    2

                      15
                     /  \
                    6    9
                        / \
                       3    6
                           / \
                          3    3
                         / \
                        1    2
```

ISSUE 8 (ADDED Sat, oct 24, 2:36pm)
Assignment of popped nodes to branches of new joint node
SOLUTION
When you pop to nodes, you are effectively popping the smallest, and the next smaller
Place the smaller in the left branch of the newly created tree

CALCULATION 1
How to get the number of relevant bits that spill over to the last byte
sum(For every leaf in tree: frequency of represented char * distance to root node)