Huffman Coding

Project 5

Huffman Steps

1. Count the number of occurrences of each character in the file. Keep track of these counts in a count array.

2. Create a binary tree consisting of just a leaf for each character with a nonzero count. The data for the tree is the character’s count. Also, create a tree with a zero count for the end of file marker. Insert all trees into a priority queue, and also store each node in an array similar to the count array.

Huffman Steps

3. While the priority queue has more than one tree in it, remove the two lowest priority trees. Create a new tree with each of these trees as subtrees. The count for the new tree is the sum of the counts of the two subtrees.

Insert the new tree into the priority queue.

4. Eventually, there will be only one tree remaining in the priority queue. This is the Huffman tree. Incidentally,the data at the root of this tree should equal the number of characters in the file.

5. Print out information about all the characters and call pack()

Tips/Hints

- Include “pack.h”
  - This is where your node struct is defined.
  - You will be calling functions at the end of your huffman program to do the encoding, you are simply creating the tree/nodes that pack will use.
  - Call pack() at the end of your main!
    - Args are (input,output,nodes array)

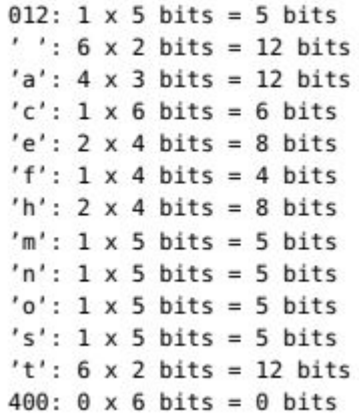
- Your nodes and count arrays need to contain 257 entries, with the last entry being the end of file character.

- Besides your main function, it is helpful to have three utility functions.
  - Depth: returns the depth of a given node
  - Node compare: comparison function for two given nodes that you pass to your createQueue
  - Make node (two purposes): 1. Create a new node with the given data. 2. Create a new node with the given data and set the parent of the left and right node passed in equal to the newly created node.

Testing

- The goal is to compress your input file into the output file, and test it by using gzip to unzip back to your original input file.

- Print out your nodes to match the lab pdf like so
  - Use the isprint function to tell if a character is printable or not, and print either the corresponding 3 digit octal value for non printable characters or the character in single quotes for printable characters.
  - The formula on the right is
    count x depth = number of bits
  - Note your numbers may be a bit different, but they should add up to around the same number of bits.



```
012: 1 x 5 bits = 5 bits
' ': 6 x 2 bits = 12 bits
'a': 4 x 3 bits = 12 bits
'c': 1 x 6 bits = 6 bits
'e': 2 x 4 bits = 8 bits
'f': 1 x 4 bits = 4 bits
'h': 2 x 4 bits = 8 bits
'm': 1 x 5 bits = 5 bits
'n': 1 x 5 bits = 5 bits
'o': 1 x 5 bits = 5 bits
's': 1 x 5 bits = 5 bits
't': 6 x 2 bits = 12 bits
400: 0 x 6 bits = 0 bits
```

Submission

- Submission deadline:
  - Sunday, May 30th at 11:59 pm
  - 10% off every 24 hrs after deadline
  - No submission will be accepted after Wed, June 2nd at 11:59 pm
- Demo deadline:
  - Lab section next week
  - No demo will be accepted in TA's office hours after your lab section next week
- File Submission:
  - Both **tar file or zip file** will be accepted
    - tar -czvf project5.tar folder_path
    - folder_path is the directory of the folder that contains both pqueue.c and huffman.c