CS340 Winter 1996
Ming Li

Assignment 1, Due Date: Jan. 24

1. Do Problems 17 and 18, page 68 in the textbook.

2. Do problems 21 and 22, page 39 in textbook.

3. (Programming) Implement the Huffman code algorithm given in class. Your program should take a file as input, where each line consists of a letter followed by a real number between 0 and 1 giving the frequency of occurrence of that letter. The frequencies should sum to 1. Your program should create the Huffman tree, and then print out, for each letter, the binary codeword corresponding to each letter. You can use as a model the pseudocode on page 340 of [Cormen, Leiserson, and Rivest: "*Introduction to Algorithms*, MIT Press] (reserved in DC library). Print out the codewords for each letter in the same order the letters appear in the input.

   Next, your program should compute the cost of the tree using the formula:

   $$C(T) = \sum_{c \in \Sigma} f(c) d_T(c),$$

   where $f(c)$ denotes the frequency of c in the file and $d_T(c)$ denotes the depth of c's leaf in the tree. Note that $d_T(c)$ is also the length of the codeword for character c. Compare this to the Shannon entropy of the letter frequencies:

   $$-\sum_{j=1}^{k} p_j \log_2 p_j$$

   where p_j is the frequency of occurrence of letter j.

   Finally, your program should then allow the user to type in strings of 0's and 1's, one at a time. Your program will then translate these binary codewords to the strings of letters they represent. A blank line is the instruction to terminate the program.

   For example, if the input to the program was the file:

   ```
   a .17
   b .14
   c .41
   d .09
   e .19
   ```

   Then the first part of the output would look like:

```
a: 110
b: 101
c: 0
d: 100
e: 111
```

The next part of the output would look like:

```
The cost of the tree is: 2.18
The Shannon entropy is : 2.127
```

Finally, the last part of the output might look like:

```
Enter codewords: 101111110100
The translation is: bead

Enter codewords:
```

Finally, test your program by running it on the following data for English letter frequencies:

```
a 0.07573
b 0.01454
c 0.04182
d 0.03305
e 0.10946
f 0.0234
g 0.01911
h 0.0389
i 0.07746
j 0.00276
k 0.00484
l 0.0391
m 0.03045
n 0.07033
o 0.07601
p 0.0309
q 0.00281
r 0.06776
```

```
s 0.07314
t 0.09273
u 0.029
v 0.01036
w 0.01455
x 0.0039
y 0.01651
z 0.00138
```

Some implementation hints: you are not required to implement a priority queue in this program. Instead, simply use a linked list or array, and find the smallest key by sequential search.

If using C, don't forget to use the math library to get the log function by including a reference to math.h, and compiling with the -lm option.