Standard Huffman Coding

Description

You are required to take an input sentence (one line) to be read from a file and compress it using Standard Huffman, save the compressed output along with the dictionary to the file (you will need the dictionary for decompression, as an input), then take that compressed output and decompress.

Grading

10 grades to be scaled later composed of (7) compression, (3) decompression. GUI (using things like text fields and of course buttons, etc. You still need to read and write to files) is bonus. (-2) grade for not using files, or less if the file read/write in the submitted code is incomplete. Saving the dictionary is part of the compression (you can not pass it to the decompression, the decompression part needs to read it from the saved file, or input if you plan to let go of a part of the file grades).

Early submission is a bonus (you get weeks/labs to submit it, the second one is the last chance). And late submissions or submissions in C++ is an automatic (0).

Compression

Input: An English sentence (remember that spaces count as a character)

Output:

- Show the count of each character in the **console** (only show count of characters that exist in the sentence i.e., characters with count > 0).
- Save the dictionary containing each letter and its corresponding binary string in a **file**.
- Save the compressed binary string (results from translating each character into its Huffman coding) in a **file**.
- Show in the console the original sentence bit size / compress bit size (compression ratio).

NOTE: You are not expected to write and read bits (actual bits (1/8 byte)). You can use regular '0' and '1' characters to form the output. This will be both easier to compare, trace and debug (for you in case you get stuck). If you choose to write bits, note that it will be your burden to figure out how to write them and read them.
Also, you can either write the dictionary and compressed data separately (each in their own file) or write them in one file.

Example (the examples are just for illustration, how you display things is up to you).
input1: The quick brown fox jumps over the lazy dog

Output1:

Dictionary: 000 (space character encoding), w 001000, x 001001, t 001010, v 001011, u 00110, y 001110, z 001111, h 01000, r 01001, e 0101, o 011, l 10000, m 10001, j 10010, k 10011, q 10100, s 10101, n 10110, p 10111, b 11000, c 11001, T 11010, a 11011, g 11100, i 11101, d 11110, f 11111

Compressed1:

11010010000101000101000011011101110011001100011000010010110010001011000011111011001
0010001001000110100011011110101000011001011010101001000001010010000101000100001101
1001111001110000111100111100

Input2 (lecture's example):

EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEETTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
OOOOOIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIINNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNN
NSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR RRHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCUUUUUUUUUUUUUUUUUUUUUUUUUU

Output2:

Dictionary2: A 110, O 111, I 0000, N 0001, S 0010, R 0011, E 010, C 01100, U 01101, H 0111, T 100, L 1010, D 1011

Compressed2:

010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010
010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010010
010010010010010010010010010010010010010010010010010100100100100100100100100100100100100100100100100100100100100100100100100100100100
100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100100
100100100100100100100100100100100100110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110
110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110110111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001
000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010
001000100010010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000
100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100011001100110011001100110011001
1001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
001100110011001100110011001100110011001100110011001101110111011101110111011101110111011101110111011101110111011101110111011101110
111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110101010101010101010101010101010
101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101011101110111011101
110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011011000110001
100011000110001100011000110001100011000110001100011000110001100011000110001100011000110001100011000110001100011000110001100010101101
011010110101101011010110101101011010110101101011010110101101011010110101101011010110101101011010110101101

OR

Output3 (variation of input2's, essentially the same length):

Dictionary3: O 000, A 001, D 0100, L 0101, T 011, H 1000, U 10010, C 10011, E 101, R 1100, S 1101, N 1110, I 1111

Compressed3:

101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101
101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101101
101101101101101101101101101101101101101101101101101101101101011011011011011011011011011011011011011011011011011011011011011011011011
011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011011
011011011011011011011011011011011011011011011001001001001001001001001001001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001001000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000011111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110
111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101
110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111
0111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110111011101110011001100110011001100110
011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100
110011001100110011001100110011001100110011001100110010001000100010001000100010001000100010001000100010001000100010001000100010001
000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010000101010101010101010101010101010101
0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010001000100010
001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100010001000100100111001110
0111001110011100111001110011100111001110011100111001110011100111001110011100111001110011100111001110011100111001110011100111001010010
100101001010010100101001010010100101001010010100101001010010100101001010010100101001010010100101001010010

(Note that all the above examples in the dictionary can have the 0, 1 exchanged or different depending on the implementation, but the lengths should be the same. Also, how you give the codes can change the middle 0's and 1's but remember that the length DOESN'T change. Unless you have something wrong. If you are confused with example 3, just go along with example 1.)

Decompression

Input: The dictionary and compressed binary string.

Output: The original sentence

Example:

Input:

Dictionary: 000 (space character encoding), w 001000, x 001001, t 001010, v 001011, u 00110, y 001110, z 001111, h 01000, r 01001, e 0101, o 011, l 10000, m 10001, j 10010, k 10011, q 10100, s 10101, n 10110, p 10111, b 11000, c 11001, T 11010, a 11011, g 11100, i 11101, d 11110, f 11111

Compressed Binary String:

1101001000010100010100001101110111001100110001100001001011001000101100001111101100100100010010001101000110111101010000110010110101010010000010100100001010001000011011001111001110000111100111110 0

Output: The quick brown fox jumps over the lazy dog

Extra:

1- You can use this line of code:

System.getProperty("user.home") + "/Desktop"

To get a string that points to the current PC's desktop (this way it will point to the desktop on any PC and if moved to another PC's desktop it will be same location without changing anything in the code).

2- For the standard Huffman you'll first need to count the characters (to get each one's entropy), then figure out how to group them and distribute the code.