LOSSLESS SOURCE CODING/ENTROPY

SECTION 1 QUESTION 1 BACKGROUND

Suppose these 7 symbols appear with the following probabilities:

Symbol	Probability
A	0.25
E	0.02
H	0.1
K	0.2
S	0.05
T	0.03
U	0.35

SECTION 1 QUESTION 1 PART A (2/2 points)

What is the entropy of the symbols, assuming the probabilities given above? Give your answer to two significant digits (e.g. 1.00).

2.31 ✓ **Answer:** 2.31

2.31

EXPLANATION

Compute the entropy according to the formula

$$H = -\sum_{i=1}^{7} p[symbol_i] log_2(p[symbol_i])$$

You have used 1 of 1 submissions

SECTION 1 QUESTION 1 PART B (2/2 points)

Find a Huffman code for these symbols.

How long is the codeword for each symbol? Please select the correct answer.

- [✓ $T\ 5\ E\ 5\ S\ 4\ H\ 3\ A\ 2\ K\ 2\ U\ 2$] ✓
- $E\ 6\ T\ 5\ H\ 4\ S\ 4\ K\ 3\ A\ 2\ U\ 1$
- $T\ 5\ E\ 5\ S\ 4\ H\ 3\ A\ 2\ K\ 2\ U\ 1$
- $E\ 6\ T\ 5\ S\ 4\ H\ 3\ K\ 3\ A\ 2\ U\ 2$

You have used 1 of 1 submissions

SECTION 1 QUESTION 1 PART C (2/2 points)

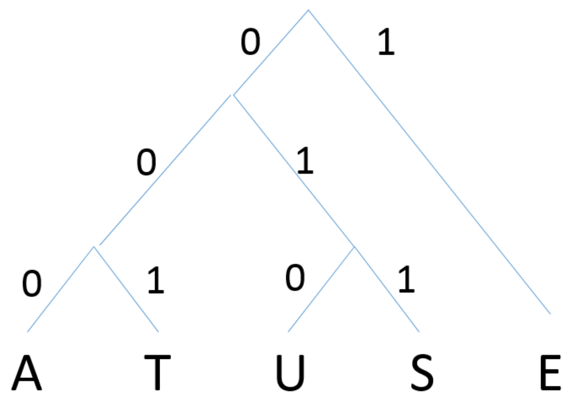
Find the average code length assuming the Huffman code is used to encode these symbols.

2.35 ✓ **Answer:** 2.35

2.35

You have used 1 of 1 submissions

SECTION 1 QUESTION 2 (2/2 points)



Suppose that the 5 symbols are encoded using the above tree, find the symbol sequence corresponding to this bit stream:
0111101000001010011001

Please input the corresponding characters below:

SEEUATUST ✓ **Answer:** SEEUATUST

You have used 1 of 1 submissions

SECTION 1 MATLAB QUESTION (3 points possible)

The goal of this task is to encode an English text using the Huffman code. The text, which is stored inside the variable **text_code**, is represented using the 8-bit ASCII code, where each symbol is described by 8 bits. For example, the letter "a" is represented by the binary number "01100001", which corresponds to the decimal number 97. In an English text some characters are more likely to be used than others, hence the message can be efficiently encoded using the Huffman code. Your task is to analyze the text, find the most used symbols and create the dictionary. As in Lab 1, instead of encoding every character with the Huffman code, we will use a hybrid code: most likely characters will be represented with the Huffman code and the others by an escape code followed by their ASCII code.

The initial MATLAB code loads the input text from the file "G4jHRdGpVaY.txt" and stores it inside the variable **text_code**. This variable is a vector of decimal integers, each lying between 0 and 255 and representing the ASCII code of the corresponding character in the input text.

The code then encodes the text using the dictionary specified by the variables **dict** and **code**. The variable **code** contains the decimal values of the ASCII codes of the N most common symbols in the text. The cell array

dict contains N+1 elements. The first N are the binary vectors for the codewords of the corresponding symbols in **code**. The last one is for the escape code. Initially, N = 0, so the dictionary contains only the escape code. All characters in the text are encoded by the escape code [0] followed by the 8 bit ASCII code of that character. The resulting bitstream is stored in the variable **huffman**.

The code then computes the length of the ASCII and the length of the encoding using the dictionary. The length of the ASCII code is 79,632, since the number of characters in the text is 9,954. The length of the bitstream **huffman** is 89,586, since each character is encoded with nine bits (0 plus the 8 bit ASCII). Finally, the code decodes the bitstream in **huffman** using the dictionary and compares it with the input text. If the dictionary is valid, the two texts will be equal.

Your first task is to find the probabilities of each of the symbols (decimal numbers) in the vector **text_code** (e.g. using the MATLAB function histogram). You can find the most common symbols by sorting this histogram. Find the smallest value of N, such that the N most common symbols account for more than 70% of symbols in **text_code**. Store the N most common symbols inside the variable **code**.

Your next task is to find the Huffman dictionary that encodes these N symbols as well as the escape sequence. Store this dictionary in the cell array **dict**. Note that the probability of the escape sequence is the remainder of the probability not accounted for by the N symbols (i.e. the probability that one of the N symbols does not occur). The total probability of the N symbols and the escape sequence should sum to one.

If your dictionary is correct, then the length of the Huffman encoding should be 54,648, which is less than that of the ASCII encoding. You are graded only on the length of the dictionary (i.e. whether you find N correctly) and the dictionary itself (i.e. whether the lengths of the codewords are correct and the dictionary can be used to encode and decode the text correctly). You are free to use any of the functions used in the previous labs.

Modify the code between the lines

% % % % Revise the following code % % % %

and

% % % % Do not change the code below % % % %

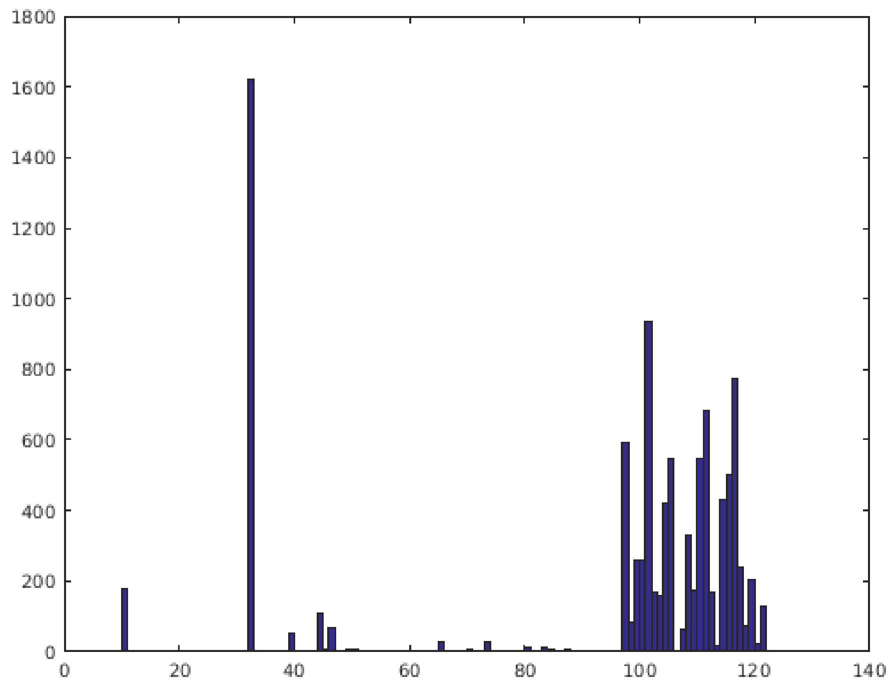
Please, do not change other parts of the code.

```
% read the text file (1 byte for each character)
fid = fopen('G4jHRdGpVaY.txt');
text_code = fread(fid,inf,'uchar');
fclose(fid);

% % % % Revise the following code % % % %
% run-length encode
%run_length_code = runlength_encode(text_code);
% convert the binary array into an decimal array of runs
%runs = bin2decArray(run_length_code);
%disp(run_length_code);
%disp(runs);
% huffman code
% compute the probability of the run lengths .
%rlen_list = [1 25 50 75 100 125 150 175 200 225 255];
%rlen_list_numValues = [0 0 0 0 0 0 0 0 0 0 0];
```

Incorrect

```
    N =10;
    % get the histogram
    code_words =[0:1:255];
    hist = histogram(text_code,code_words,'Normalization','probabi
lity');
    prob = hist.Values;
    % sort the histogram
    [values, id_codes] = sort(prob, 'descend');

    display(['Probability covered by the first ' num2str(N) ' char
acters: ' num2str(sum(values(1:N)))]);
    display(['Most likely characters: "' char(code_words(id_codes(
1:N))) '"']);
    % get the huffman code (code N+1 = escape code)
    dict = {[0 0 0],[1 1 0],[0 0 1 0],[0 0 1 1],[1 0 0 0],[1 0 0 1
],[1 0 1 0],[1 0 1 1],[1 1 1 0],[1 1 1 1],[0 1]};
    code = code_words(id_codes(1:N));
```

Figure 1



```
      9954

    125

   Columns 1 through 17

     0.0182    0.1627    0.0002    0.0052    0.0106    0.0009    0.0069    0.0002
 0.0005    0.0006    0.0004    0.0003    0.0001    0.0001    0.0025    0.0001    0.0
001

   Columns 18 through 34

     0.0005    0.0003    0.0027    0.0001    0.0002    0.0002    0.0003    0.0011
 0.0002    0.0011    0.0005    0.0001    0.0008    0.0002    0.0595    0.0082    0.0
260

   Columns 35 through 51

     0.0260    0.0940    0.0167    0.0161    0.0424    0.0551    0.0002    0.0065
 0.0332    0.0173    0.0552    0.0687    0.0170    0.0017    0.0431    0.0505    0.0
777

   Columns 52 through 57

     0.0238    0.0071    0.0203    0.0023    0.0131    0.0003

   Columns 1 through 29
```

```
  10    32    37    39    44    45    46    48    49    50    51    52    54    63    65    6
7    69    70    72    73    75    77    78    79    80    81    83    84    86

 Columns 30 through 57

  87    89    97    98    99   100   101   102   103   104   105   106   107   108   1
09   110   111   112   113   114   115   116   117   118   119   120   121   122
```

- The length of the variable "dict" is incorrect.

You have used 2 of 5 submissions