1) My sentence is **"TODAY IS THE LONGEST DAY OF THE SPRING SEMESTER IN BILKENT"** and the PMF of the occurrence of characters is;

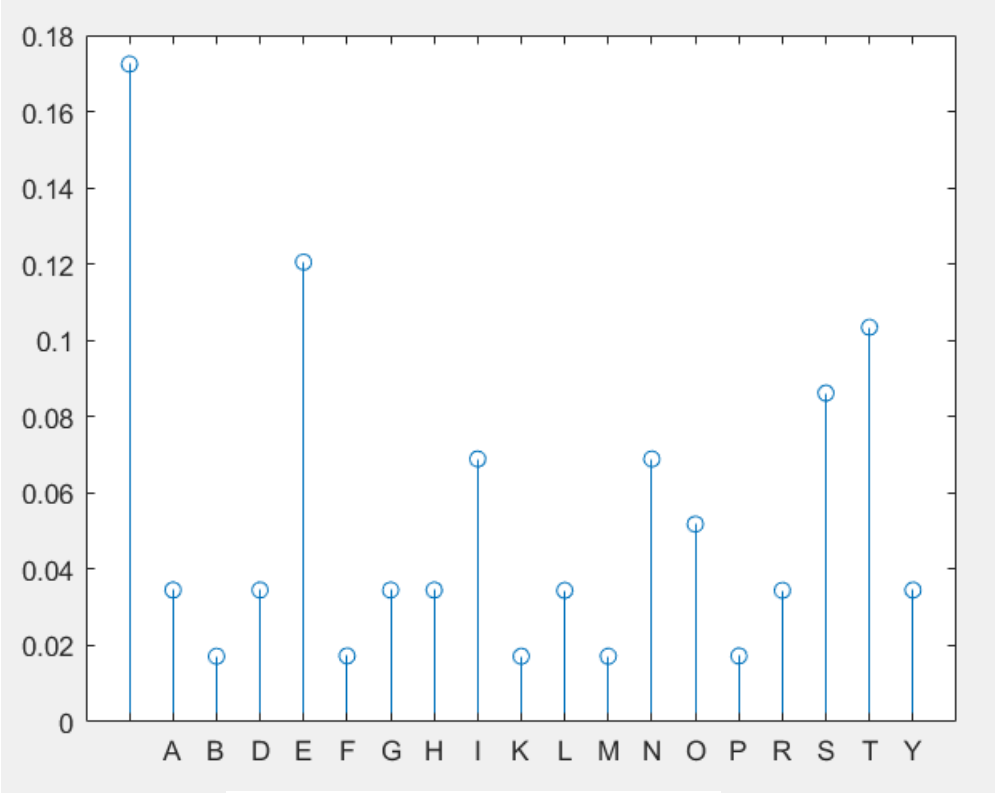

Figure 4: PMF of the input sentence

According to this PMF, Huffman coding is implemented and codewords are;

' :000', '**E**:100', '**T**:110', '**S**:111', '**I**:0110', '**N**:0111', '**O**:1010', '**A**:00100', '**G**:00101', '**H**:00110', '**D**:00111', '**Y**:01000', '**R**:01010', '**L**:01011', '**F**:10111', '**P**:010010', '**B**:010011', '**M**:101100', '**K**:101101'

The average codeword length is 3.9139 and the entropy is 3.8797. In other words, Huffman encoding algorithm is close to the best possibility. Without coding, since we have 19 characters and 4.24 bits per word would be used. Thus, by Huffman coding we saved approximately 0.25 bits for this specific example.

In my Huffman encoding algorithm, first I take a char array which includes the letters in the sentence and turned it string array and a double array which includes probabilities of the letters, in both arrays' same indexes are belong to each other. Secondly, probabilities and characters are sorted from highest to the lowest in arrays. Thirdly, 2-d arrays are built for Huffman coding. In the first row letters are sorted according to their probability and last two elements are combined, both as a string and probabilities, and in the second row elements are sorted again including the new element which is a combination of last two elements of the first row. This process is done until there are only 2 elements left. Lastly, every letter is

checked in the 2-d array and if the element occurs at the last column of the row codeword ‘1’ is given if it occurs at the column before the last one codeword ‘0’ is given.

H	D	Y	R	L	F	P	B	M	K
H	D	Y	R	L	MK	F	P	B	0
H	D	Y	PB	R	L	MK	F	0	0

Figure 5: Example of Encoding Algorithm

2) For encoder part, in a 2-d array every letter’s codewords are kept as strings and if a codeword occurs in the given input sequence, algorithm checks the codeword’s letter from the 2-d array and displays it. There is an example in *Figure 6* for Huffman decoder for 1 input sequence but of course in the MATLAB assignment it is done for more than 1 million inputs.

```
input_sequence = ['1101010001110010001000000011011100011000110100000010111011' ...
'0011100101100111110000001110010001000000101010111000110001101000001110100010' ...
'0101001100111001010001111001011001001111101000101000001100111000010011011000' ...
'10111011011000111110'];
disp("Decoded input is: " + huffman_decoder(input_sequence, code_array));
```

Figure 6: Example Input for Decoder

```
Decoded input is: TODAY IS THE LONGEST DAY OF THE SPRING SEMESTER IN BILKENT
```

Figure 7: Result of the Decoder