```java
import java.util.HashMap;
import java.util.PriorityQueue;
import java.util.Scanner;

public class HuffmanCoding {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("Enter 3 lines to be encoded.\n");
        StringBuilder txt = new StringBuilder();

        String [] t = new String [3];

        //read 3 lines
        for (int i = 0; i < 3; i++) {
            t[i] = scan.nextLine();
            txt.append(t[i]);
            txt.append(" ");
        }

        String text = txt.toString();
        System.out.println("\nThe text is :\n" + text + "\n");

        HuffmanCoding.variableLengthEncoding(text);
        HuffmanCoding.fixedLengthEncoding(text);

        //Part 3
        t = new String[3];
        int countArr [][] = new int [3][2];

        for(int i = 0; i < 3; i++){
            System.out.println("Enter paragraph " + (i+1) + ":");
            t[i] = scan.nextLine();
            countArr[i][0] = variableLengthEncodeCount(t[i]);
            countArr[i][1] = fixedLengthEncodeCount(t[i]);
        }

        System.out.println("\nVariable Length Encoding Count\t\tFixed Length Encoding Count ");
        for(int i = 0; i < 3; i++){
            System.out.println("\t   " + countArr[i][0] + "\t\t\t\t\t   " + countArr[i][1]);
        }

        scan.close();

    }

    public static int variableLengthEncodeCount(String text){
```

```
49          HashMap <Character, Integer> freq = getFreq(text);
50          Node root = huffman(freq);
51          HashMap <Character, String> varHuffmap = new HashMap<>();
52          HashMap <String, Character> varRevHuffmap = new HashMap<>();
53          varCharEncode(root, "", varHuffmap);
54
55          //reverse of huffmap
56          for(char c : varHuffmap.keySet()){
57              varRevHuffmap.put(varHuffmap.get(c),c);
58          }
59
60          //Encoding the text
61          String varEncode = encodeText(text, varHuffmap);
62          return varEncode.length();
63      }
64
65      public static int fixedLengthEncodeCount(String text) {
66          HashMap <Character, String> fixHuffmap = new HashMap<>();
67          HashMap <String, Character> fixRevHuffmap = new HashMap<>();
68
69          //char to binary string
70          for(char c : text.toCharArray()){
71              fixHuffmap.put(c, charToBinary(c));
72          }
73
74          //reverse huffmap
75          for(char c : fixHuffmap.keySet()){
76              fixRevHuffmap.put(fixHuffmap.get(c),c);
77          }
78
79          String fixEncode = encodeText(text, fixHuffmap);
80          return fixEncode.length();
81      }
82
83      public static void variableLengthEncoding(String text) {
84          //frequencies
85          HashMap <Character, Integer> freq = getFreq(text);
86
87          //huffman tree
88          Node root = huffman(freq);
89
90          //Var length
91          HashMap <Character, String> varHuffmap = new HashMap<>();
92          HashMap <String, Character> varRevHuffmap = new HashMap<>();
93          //encoding of each char
94          varCharEncode(root, "", varHuffmap);
95
96          //reverse of huffmap
97          for(char c : varHuffmap.keySet()){
98              varRevHuffmap.put(varHuffmap.get(c),c);
```

```
99          }
100
101         System.out.println("Frequencies of each character:\n" + freq + "\n");
102         System.out.println("Variable length code mapping for each character:");
103
104         for(char c : varHuffmap.keySet()){
105             System.out.println(c + " " + varHuffmap.get(c));
106         }
107         System.out.println();
108
109         //Encoding the text
110         String varEncode = encodeText(text, varHuffmap);
111
112         System.out.println("Compressed encoded code: \nVariable length encoding: \n"
    + varEncode + "\n");
113
114         //decoding the text
115         String decoString = decode(varEncode, varRevHuffmap);
116         System.out.println("Original text after decoding: \n" + decoString +"\n");
117
118     }
119
120     public static void fixedLengthEncoding(String text) {
121         //Fixed Length
122
123         HashMap <Character, String> fixHuffmap = new HashMap<>();
124         HashMap <String, Character> fixRevHuffmap = new HashMap<>();
125
126         //char to binary string
127         for(char c : text.toCharArray()){
128             fixHuffmap.put(c, charToBinary(c));
129         }
130
131         //reverse huffmap
132         for(char c : fixHuffmap.keySet()){
133             fixRevHuffmap.put(fixHuffmap.get(c),c);
134         }
135
136         System.out.println("Fixed length code mapping for each character:");
137
138         for(char c : fixHuffmap.keySet()){
139             System.out.println(c + " " + fixHuffmap.get(c));
140         }
141         System.out.println();
142
143         //Encoding the text
144         String fixEncode = encodeText(text, fixHuffmap);
145
146         System.out.println("Compressed encoded code: \nFixed length encoding: \n" +
    fixEncode + "\n");
147
```

```
148         //decoding the text
149         String decoString = decode(fixEncode, fixRevHuffmap);
150         System.out.println("Original text after decoding: \n" + decoString +"\n");
151     }
152 
153 
154     public static HashMap <Character, Integer> getFreq(String text){
155         HashMap <Character, Integer> freq = new HashMap<>();
156         for(char c : text.toCharArray()){
157             freq.put(c,freq.getOrDefault(c, 0) + 1);
158         }
159         return freq;
160     }
161 
162     public static String encodeText(String txt, HashMap <Character, String> huffmap){
163         StringBuilder encodeBuilder = new StringBuilder();
164         for(char c : txt.toString().toCharArray()){
165             encodeBuilder.append(huffmap.get(c));
166         }
167         return encodeBuilder.toString();
168     }
169 
170     public static String charToBinary(char c){
171         String temp = Integer.toBinaryString(c);
172         if(temp.length() < 8){
173             int length = temp.length();
174             for(int i = 0; i < (8 - length); i++){
175                 temp = "0" + temp;
176             }
177         }
178         return temp;
179     }
180 
181     public static boolean isLeaf(Node root){
182         return root.left == null && root.right == null;
183     }
184 
185     public static void varCharEncode(Node root, String s, HashMap <Character, String>
    huffmap) {
186         if(root == null) return;
187         if(isLeaf(root)) huffmap.put(root.c,s.length() > 0 ? s : "1");
188         varCharEncode(root.left, s + '0', huffmap);
189         varCharEncode(root.right, s + '1', huffmap);
190     }
191 
192     public static String decode(String encodeBuilder, HashMap <String, Character>
    revHuffmap){
193         StringBuilder sb = new StringBuilder(), temp = new StringBuilder();
194         char [] charArr = encodeBuilder.toCharArray();
195         for(int i = 0; i < encodeBuilder.length(); i++){
```

```
            temp.append(charArr[i]);
            if(revHuffmap.containsKey(temp.toString())){
                sb.append(revHuffmap.get(temp.toString()));
                temp = new StringBuilder();
            }

        }

        return sb.toString();
    }


    public static Node huffman(HashMap <Character,Integer> freq) {
        PriorityQueue <Node> q = new PriorityQueue<>( (a,b) -> a.freq - b.freq);

        for (char c : freq.keySet()) {
            Node tempNode = new Node(c, freq.get(c));
            q.add(tempNode);
        }

        while(q.size() > 1){
            Node left = q.poll();
            Node right = q.poll();
            Node newNode = new Node(left.freq + right.freq);
            newNode.left = left;
            newNode.right = right;
            q.add(newNode);
        }

        return q.poll();
    }

}

class Node{
    char c;
    Node left;
    Node right;
    int freq;
    Node(char c, int freq){
        this.c = c;
        this.freq = freq;
    }
    Node(int freq){
        this.freq = freq;
    }

}
```

```
rohitkrishnanvidyasagar@Rohits-MacBook-Air assignment-7-huffman % java HuffmanCoding
Enter 3 lines to be encoded.

As the semester draws to a close, we look back and reflect
On all the things that have happened
In order to make sense of these events.

The text is :
As the semester draws to a close, we look back and reflect On all the things that have happened In order to make sense of these events.

Frequencies of each character:
{A=1, I=1, O=1,  =27, a=9, b=1, c=3, d=4, e=20, f=2, g=1, h=7, i=1, k=3, l=5, ,=1, m=2, n=7, .=1, o=7, p=2, r=5, s=10, t=11, v=2, w=2}

Variable length code mapping for each character:
A 1000110
I 1000111
O 0111101
  00
a 1001
b 1000010
c 110001
d 111111
e 101
f 011101
g 0111100
h 0100
i 1000011
k 111110
, 1100000
l 11110
m 100000
n 0110
. 1100001
o 0101
p 011100
r 11001
s 1101
t 1110
v 011111
w 100010

Compressed encoded code:
Variable length encoding:
10001101101001110010010100110110110000010111011110101110010011111111001100110001011010011100101001001001100011111001011101101110000000100010101001111001010101111110001000010100111000111111000100101101111110
01100110101110111110101110001111000011110101100010011111011111000111001001010011100100100001101100111100110100111001001001111000010010010111111010001001001011100011100101011010111111100100011101100001011100 1
111111101110010011100101001000001001111110101001101101011011011010001010111010011100100101110110100101011111101011011101101110000100

Original text after decoding:
As the semester draws to a close, we look back and reflect On all the things that have happened In order to make sense of these events.

Fixed length code mapping for each character:
A 01000001
I 01001001
O 01001111
  00100000
a 01100001
b 01100010
c 01100011
d 01100100
e 01100101
f 01100110
g 01100111
h 01101000
i 01101001
```

```
t 1110
v 011111
w 100010

Compressed encoded code:
Variable length encoding:
10001101101001110010010100110110110000010111011110101110010011111111001100110001011010011100101001001001100011111001011101101110000000100010101001111001010101011111100010000101001110001111110001001011011111110
01100110101110111110101110001111000011110101100010011111011110001110010010100111001001000011011001111001101001110010010011110000100100101111110100010010010111000111001010110101111111001000111011000010111001
11111110111001001110010100100000100111111010100110110101101101101000101011101001110010010111011010010101111110101101110110111 0000100

Original text after decoding:
As the semester draws to a close, we look back and reflect On all the things that have happened In order to make sense of these events.

Fixed length code mapping for each character:
A 01000001
I 01001001
O 01001111
  00100000
a 01100001
b 01100010
c 01100011
d 01100100
e 01100101
f 01100110
g 01100111
h 01101000
i 01101001
k 01101011
l 01101100
, 00101100
m 01101101
n 01101110
. 00101110
o 01101111
p 01110000
r 01110010
s 01110011
t 01110100
v 01110110
w 01110111

Compressed encoded code:
Fixed length encoding:
010000010111001100100000011101000110100001100101001000000111001101100101011011010110010101110011011101000110010101110010001000000110010001110010011000010111011101110011001000000111010001101111001000000110000
010010000000110001101101100011011110111001101100101001011000010000000111011101100101001000000110110001101111011011110110101100100000011000100110000101100011011010110010000000110000101101110011001000010000000111
0010011001010110011001101100011001010110001101110100001000000100111101101110001000000110000101101100011011000010000000111010001101000011001010010000000111010001101000011010010110111001100111011100110010000001
110100011010000110000101110100001000000110100001100001011101100110010100100000011010000110000101110000011100000110010101101110011001010110010000100000010010010110111000100000011011110111001001100100011001010
0111001000100000011101000110111100100000011011010110000101101011011001010010000000111001101100101011011100111001101100101001000000110111101100110001000000111010001101000011001010111001101100101001000000110010
0101110110011001010110111001110100011100110010111000100000
Original text after decoding:
As the semester draws to a close, we look back and reflect On all the things that have happened In order to make sense of these events.

Enter paragraph 1:
There will be a new dawn
Enter paragraph 2:
In the ear of computing!
Enter paragraph 3:
And we will be here to witness it

Variable Length Encoding Count          Fixed Length Encoding Count
           81                                     192
           96                                     192
           116                                    264
rohitkrishnanvidyasagar@Rohits-MacBook-Air assignment-7-huffman %
```