

```

1  // Comparing the efficiency of various keyboard layouts
2  // Based on user input, the amount of distance that needs to be travelled to press each key is calculated, provided all fingers are used
3  // The layouts in which less distance is travelled is most efficient.
4  // Currently there are qwerty, dvorak and colemak layouts which are compared.
5
6  import java.util.Scanner;
7
8  public class keyboardAnalyser {
9      public static void main(String[] args) {
10         Scanner sc = new Scanner(System.in);
11         System.out.println("Enter sentence: ");
12         String sentence = sc.nextLine();
13         sc.close();
14         sentence = sentence.trim().concat(" ");
15         System.out.println("Distance moved by fingers to type above sentence: ");
16         System.out.println("\nQwerty: ");
17         wordFingerDistance(sentence, "QWERTY");
18         System.out.println("\nDvorak: ");
19         wordDistance(sentence, "DVORAK");
20         wordFingerDistance(sentence, "DVORAK");
21         System.out.println("\nColemak: ");
22         wordDistance(sentence, "COLEMAK");
23         wordFingerDistance(sentence, "COLEMAK");
24     }
25
26     // Finding the finger required to press a given letter for all layouts
27     static int findFingerQWERTY(char a) {
28         if ("qaz".contains(Character.toString(a))) return 0;
29         if ("wsx".contains(Character.toString(a))) return 1;
30         if ("edc".contains(Character.toString(a))) return 2;
31         if ("rfvtgb".contains(Character.toString(a))) return 3;
32         if ("yhnujm".contains(Character.toString(a))) return 4;
33         if ("ik".contains(Character.toString(a))) return 5;
34         if ("ol".contains(Character.toString(a))) return 6;
35         if ("p".contains(Character.toString(a))) return 7;
36         else return 8;
37     }
38
39     static int findFingerDVORAK(char a) {
40         if ("a".contains(Character.toString(a))) return 0;
41         if ("oq".contains(Character.toString(a))) return 1;
42         if ("ej".contains(Character.toString(a))) return 2;
43         if ("pukyix".contains(Character.toString(a))) return 3;
44         if ("fdbghm".contains(Character.toString(a))) return 4;
45         if ("ctw".contains(Character.toString(a))) return 5;
46         if ("rnv".contains(Character.toString(a))) return 6;
47         if ("lsz".contains(Character.toString(a))) return 7;
48         else return 8;
49     }
50
51     static int findFingerCOLEMAK(char a) {
52         if ("qaz".contains(Character.toString(a))) return 0;
53         if ("wrx".contains(Character.toString(a))) return 1;
54         if ("fsc".contains(Character.toString(a))) return 2;
55         if ("ptvgdb".contains(Character.toString(a))) return 3;
56         if ("jhklne".contains(Character.toString(a))) return 4;
57         if ("ue".contains(Character.toString(a))) return 5;
58         if ("yi".contains(Character.toString(a))) return 6;
59         if ("o".contains(Character.toString(a))) return 7;
60         else return 8;
61     }
62
63     // Finding if the character is on the home row (finger doesn't to move to press it) or not on the home row (the finger has to move to press it)
64     static int findDistanceQWERTY(int finger, int ch) {
65         int distance = 0;
66         if (finger == 0) {
67             if (ch != 'a')
68                 distance += 1;
69         }
70         if (finger == 1) {
71             if (ch != 's')
72                 distance += 1;
73         }
74         if (finger == 2) {
75             if (ch != 'd')
76                 distance += 1;
77         }
78         if (finger == 3) {
79             if (ch != 'f')
80                 distance += 1;
81         }
82         if (finger == 4) {
83             if (ch != 'j') {
84                 distance += 1;
85             }
86         }
87         if (finger == 5) {
88             if (ch != 'k') {
89                 distance += 1;
90             }
91         }
92         if (finger == 6) {
93             if (ch != 'l') {
94                 distance += 1;
95             }
96         }
97         if (finger == 7) {

```

```

98         distance += 1;
99     }
100     return distance;
101 }
102
103 static int findDistanceDVORAK(int finger, int ch) {
104     int distance = 0;
105     if (finger == 1) {
106         if (ch != 'o')
107             distance += 1;
108     }
109     if (finger == 2) {
110         if (ch != 'e')
111             distance += 1;
112     }
113     if (finger == 3) {
114         if (ch != 'u')
115             distance += 1;
116     }
117     if (finger == 4) {
118         if (ch != 'h') {
119             distance += 1;
120         }
121     }
122     if (finger == 5) {
123         if (ch != 't') {
124             distance += 1;
125         }
126     }
127     if (finger == 6) {
128         if (ch != 'n') {
129             distance += 1;
130         }
131     }
132     if (finger == 7) {
133         if (ch != 's') {
134             distance += 1;
135         }
136     }
137     return distance;
138 }
139
140 static int findDistanceCOLEMAK(int finger, int ch) {
141     int distance = 0;
142     if (finger == 0) {
143         if (ch != 'a')
144             distance += 1;
145     }
146     if (finger == 1) {
147         if (ch != 'r')
148             distance += 1;
149     }
150     if (finger == 2) {
151         if (ch != 's')
152             distance += 1;
153     }
154     if (finger == 3) {
155         if (ch != 't')
156             distance += 1;
157     }
158     if (finger == 4) {
159         if (ch != 'n')
160             distance += 1;
161     }
162     if (finger == 5) {
163         if (ch != 'e') {
164             distance += 1;
165         }
166     }
167     if (finger == 6) {
168         if (ch != 'i')
169             distance += 1;
170     }
171     return distance;
172 }
173
174 // Finding the total distance required by the fingers to press each key of the input
175 static void wordDistance(String word, String layout) {
176     int distance = 0;
177     for (int i = 0; i < word.length(); i++) {
178         char ch = word.charAt(i);
179         if (layout.equals("QWERTY"))
180             distance += findDistanceQWERTY(findFingerQWERTY(ch), ch);
181         if (layout.equals("DVORAK"))
182             distance += findDistanceDVORAK(findFingerDVORAK(ch), ch);
183         if (layout.equals("COLEMAK"))
184             distance += findDistanceCOLEMAK(findFingerCOLEMAK(ch), ch);
185     }
186     System.out.println(distance);
187 }
188
189 // Finding the distance moved by each finger separately to press each key and then the average of each
190 static void wordFingerDistance(String word, String layout) {
191     int[] distance = {0, 0, 0, 0, 0, 0, 0, 0, 0};
192     for (int i = 0; i < word.length(); i++) {
193         char ch = word.charAt(i);
194         if (layout.equals("QWERTY"))
195             distance[findFingerQWERTY(ch)] += findDistanceQWERTY(findFingerQWERTY(ch), ch);
196         if (layout.equals("DVORAK"))
197             distance[findFingerDVORAK(ch)] += findDistanceDVORAK(findFingerDVORAK(ch), ch);

```

```
198         if (layout.equals("COLEMAK"))
199             distance[findFingerCOLEMAK(ch)] += findDistanceCOLEMAK(findFingerCOLEMAK(ch), ch);
200     }
201     System.out.println("Layout: " + layout);
202     int avDistance = 0;
203     for (int i = 0; i < distance.length; i++) {
204         System.out.println("Finger " + (i + 1) + ": " + distance[i]);
205         avDistance += distance[i];
206     }
207     // Sum and Average of all fingers
208     System.out.println("Total distance: " + avDistance);
209     System.out.println("Average of all fingers: " + (avDistance / distance.length));
210 }
211 }
```