

```

1  /*
2  ****
3  *                               CEASAR
4  ****
5
6  Command format:
7
8          cesar <input.txt> <output.txt> <key> <row>
9
10 It encodes the text in <input.txt> with Cesar coding. The coding key is a
11 positive integer <key>. It encodes a sentence indicated by <row> (a positive
12 integer), between 2 consecutive full stop characters.
13 The encoded sentence is printed in <output.txt> (as a verification also the
14 encoded alphabet is printed).
15 The text in <input.txt> is saved into a 2D array where each row contains one
16 sentence (including the full stop character).
17 The correctness of the command format is checked.
18 ****
19 */
20
21 # include <stdio.h>
22 # include <string.h>
23 # include <stdlib.h>
24 # include <ctype.h>
25
26 # define MAX 1000
27
28 /* To abbreviate */
29 typedef short unsigned int S_U_INT;
30
31 /* Function Prototypes */
32 unsigned int FtoM(FILE*, char[][MAX]);
33 void CesarCoding(char[], S_U_INT, char[]);
34
35
36
37 int main(int argc, char* argv[]) {
38
39     /*
40     argv+1 contains the name of the input file.
41     argv+2 contains the name of the input file.
42     argv+3 contains the key value.
43     argv+4 contains the row to encode.
44     */
45
46     /* Check number of arguments */
47     if (argc != 5) {
48         printf("\n Incorrect Command!\n");
49         return 0;
50     }
51
52     FILE* fin;
53     unsigned int rows, sentence;
54     char m[MAX][MAX];
55
56     fin = fopen(*(argv + 1), "r");
57     rows = FtoM(fin, m);
58     sentence = atoi(*(argv + 4));

```

```

59
60     /* Check the value of <row> */
61     if (sentence > rows) {
62         printf("\nThe sentence numbers is higher than the ");
63         printf("number of sentences %s", *(argv + 1));
64         return 0;
65     }
66
67     sentence--;    /* READ BELOW */
68
69     char out[100];
70     S_U_INT key;
71
72     key = atoi(*(argv + 3));
73     strcpy(out, *(argv + 2));
74     CaesarCoding(m[sentence], key, out);
75     fclose(fin);
76 }
77
78 /* The value of 'sentence' must be decremented by 1 because the rows of 'm' have
79 indeces starting from 0 */
80
81 /*****
82 *****/
83
84
85
86 *****/
87 *           CaesarCoding
88 *****/
89 Receives a string 's' as input, an unsigned short integer 'key' and a string
90 'out' containing the name of the output file.
91 It performs the Caesar coding using the given key.
92
93 It checks the value of 'key' (must be smaller than 26). It also checks that
94 's' actually contains text.
95
96 Two strings 'a' and 'A' to contain the letters of the English alphabet (the
97 length of these strings is 26 plus one, accounting for the end of string
98 character '\0') and to treat normal and capital letters.
99 The coding alphabets are built into 'ak' and 'AK' from 'a' and 'A' using
100 'key': The first k letters in these alphabets are the last k in the normal
101 alphabet.
102
103 The indeces to address the letters in the alphabets can be calculated as
104
105 s[i] - 'a' (for non-capital letters).
106
107 The calculated index can be used in the coded alphabet to get the coded
108 character.
109
110 NB The function doesn't take into account special characters.
111 */
112
113
114
115
116

```

```
117
118 void CaesarCoding(char s[], S_U_INT key, char out[]) {
119     /* Check value of 'key' */
120     if (key > 25)
121         printf("\nKey must be smaller than 26!\n");
122
123     /* Check that 's' contains text */
124     else if ((s[0] == '.' && s[1] == '\0') || s[0] == '\0') {
125         printf("\nNo text!\n");
126     }
127     else {
128         FILE * fout;
129
130         short int j;
131         char ak[27], AK[27];
132
133         char a[] = "abcdefghijklmnopqrstuvwxyz";
134         char A[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
135
136         fout = fopen(out, "w");
137
138         j = 26 - key;
139
140         strcpy(ak, (a + j));
141         strcpy(AK, (A + j));
142         strncat(ak, a, j);
143         strncat(AK, A, j);
144
145         for (int i = 0; s[i] != '\0'; i++) {
146             if (s[i] >= 'a' && s[i] <= 'z') {
147                 j = s[i] - 'a';
148                 putchar(ak[j], fout);
149             }
150             else if (s[i] >= 'A' && s[i] <= 'Z') {
151                 j = s[i] - 'A';
152                 putchar(AK[j], fout);
153             }
154             else
155                 putchar(s[i], fout);
156         }
157
158         fprintf(fout, "\n\n%s\n%s\n", A, AK);
159         fprintf(fout, "\n%s\n%s\n\nEND.", a, ak);
160         printf("\nRead the result in %s\n", out);
161
162         fclose(fout);
163     }
164 }
165
166
167 /*****
168 *****/
169
170
171
172
173
174
```

```
175  /*
176  ****
177  *                               FtoM
178  ****
179
180  Receives a pointer to a file and a 2D array of char.
181  Copies each sentence in a row of the array and returns the number of rows that
182  have been filled, which is the number of sentences.
183
184  It ignores empty spaces at the beginning.
185
186  To determine the end of a sentence it checks that an empty space, a new line or
187  a tabulation character follows a full stop character (the latter is copied too).
188  A new line character in the text is translated into an empty space in the array.
189  If other characters follow a full stop character, it continues to copy in the
190  same sentence.
191
192  BEWARE: other types of mistakes (e.g. punctuation characters after '.') are
193  not checked.
194  */
195
196
197  unsigned int FtoM(FILE* f, char m[][MAX]) {
198      int c, i = 0, j = 0;
199
200      while ((c = getc(f)) != EOF && j < MAX) {
201          /* Initial empty spaces */
202          if (isspace(c) && j == 0)
203              continue;
204          /* Copy everything that isn't '.' or '\n' */
205          else if (c != '.' && c != '\n')
206              m[i][j++] = c;
207          else if (c == '.') {
208              m[i][j++] = c;
209              c = getc(f);
210              if (isspace(c)) {
211                  m[i++][j] = '\0';
212                  j = 0;
213              }
214              else if (c == EOF) {
215                  m[i][j] = '\0';
216                  break;
217              }
218              else
219                  m[i][j++] = c;
220          }
221          else
222              m[i][j++] = ' ';
223      }
224      return i + 1;
225  }
```