```
1
2
    * HW04 Q2
3
    * Student Name: HASAN MEN
4
5
    * Student ID : 131044009
    * Date
6
              : 15.3.15
              : This program takes an file which has crypted*
7
8
     messages and turn this codes encoded file after that
9
      solves cypher and writes plain message in plain text file*
10
    11
   #include <stdio.h>
12
13
14
   #define PLAINTEXTFILE "Files/Q2/ReceivedMessage.txt"
15
   #define ENCODEDFILE "Files/Q2/EncodedInput.txt"
16
17
   #define CRYPTEDINPUT "Files/Q2/CryptedInput.txt"
18
   /*function prototypes*/
19
20
   21
    * Gets FILE* to write file and character to decode
22
    * uses encoding table to convert encoded message to
23
24
     plain text message
    25
26
   decode_and_write_to_file(FILE *f_out_ptr, int number_of_ones);
27
28
   29
    * Gets FILE* f_in_ptr to read from encoded text file and
30
    * FILE* f_out_ptr to write message to plain text file
31
32
     return number of characters read from encoded text
    33
34
   int
   decode_message(FILE *f_in_ptr, FILE *f_out_ptr);
35
36
   37
    * Gets FILE* f_in_ptr to read from encrypted text file and *
38
    * FILE* f_out_ptr to write message to encoded file
39
40
    * return encrypted character number
    41
42
   int
43
   decrypt_message(FILE *f_in_ptr, FILE *f_out_ptr);
44
   45
    * Reads encrypted text and creates encoded and
46
47
    * plain text files
    48
49
   int
50
   main(int argc, char* argv[])
51
52
      FILE *f_plane_ptr, *f_encoded_ptr, *f_crypted_ptr;
53
54
      /* exit progtam and print error if files couldn't be opened*/
55
      if((f_crypted_ptr= fopen(CRYPTEDINPUT, "r"))==NULL ||
         (f_encoded_ptr= fopen(ENCODEDFILE,"w"))==NULL )
56
57
         if((f_crypted_ptr= fopen(CRYPTEDINPUT, "r"))==NULL)
58
59
            printf("Can't open the CRYPTEDINPUT to read.\n");
60
61
            printf("Can't open the ENCODEDFILE to write.\n");
62
      }
63
      /*files opened , countinued */
64
65
      else
66
67
          /*call decrypt_message() func. */
68
69
         decrypt_message(f_crypted_ptr, f_encoded_ptr);
70
71
         /*close files*/
72
         fclose(f_crypted_ptr);
         fclose(f_encoded_ptr);
73
74
```

```
/* exit progtam and print error if files couldn't be opened*/
75
             if((f_plane_ptr= fopen(PLAINTEXTFILE,"w"))==NULL ||
76
77
                 (f_encoded_ptr= fopen(ENCODEDFILE,"r"))==NULL )
78
79
                 if((f_plane_ptr= fopen(PLAINTEXTFILE, "w"))==NULL)
                         printf("Can't open the PLAINTEXTFILE to write\n");
80
81
82
                     printf("Can't open the ENCODEDFILE to read.\n");
83
             }
84
             /*files opened , countinued */
85
             else
86
87
88
89
             /*call decode_message() */
90
             decode_message(f_encoded_ptr, f_plane_ptr);
             printf("\n**** ENCRYPTED ****\n");
91
92
93
             /*close_files*/
             fclose(f_crypted_ptr);
94
             fclose(f_encoded_ptr);
95
96
97
         return 0;
98
99
         /*end of main */
100
101
     102
      * Gets FILE* to write file and character to decode
103
      * uses encoding table to convert encoded message to
104
105
        plain text message
106
      107
108
     decode_and_write_to_file(FILE *f_out_ptr, int number_of_ones)
109
110
111
         /* write character according to number of ones */
         switch(number_of_ones)
112
113
             case 0: fprintf(f_out_ptr,"E"); break;
114
             case 1: fprintf(f_out_ptr,"I"); break;
115
             case 2: fprintf(f_out_ptr," "); break;
116
             case 3: fprintf(f_out_ptr,"T"); break;
117
             case 4: fprintf(f_out_ptr,"C"); break;
118
             case 5: fprintf(f_out_ptr,"N"); break;
119
             case 6: fprintf(f_out_ptr,"A"); break;
case 7: fprintf(f_out_ptr,"G"); break;
case 8: fprintf(f_out_ptr,"B"); break;
case 9: fprintf(f_out_ptr,"Z"); break;
120
121
122
123
             case 10:fprintf(f_out_ptr,"H"); break;
124
             case 11:fprintf(f_out_ptr,"L"); break;
125
             case 12:fprintf(f_out_ptr,"U"); break;
126
             case 13:fprintf(f_out_ptr,"V"); break;
127
             case 14:fprintf(f_out_ptr,"R"); break;
case 15:fprintf(f_out_ptr,"S"); break;
case 16:fprintf(f_out_ptr,"Y"); break;
128
129
130
131
132
         }
133
     }
134
135
     136
137
      * Gets FILE* f_in_ptr to read from encoded text file and
      * FILE* f_out_ptr to write message to plain text file
138
      * return number of characters read from encoded text
139
140
      141
     decode_message(FILE *f_in_ptr, FILE *f_out_ptr)
142
143
     {
144
          /* Hint: While reading from encoded text file keep reading
145
                  character by character. Count the number of ones and
146
                  Call decode_and_write_to_file function when you detect 0.
          */
147
148
```

```
149
                             /* number of 1-0 readed from file */
         int counter = 0;
150
                             /* keeps 1-0 */
         char character;
151
         int number_of_ones=0;
                                /* number of ones*/
         /*end of local variables */
152
153
154
         while((fscanf(f_in_ptr,"%c",&character))!=E0F)/*read until end of file*/
155
156
                     counter++;
157
                     if(character=='1')
158
                     {
                         number_of_ones++;
159
                     }
160
                     else if(character=='0')
161
162
                         decode_and_write_to_file(f_out_ptr,number_of_ones);
163
                         number_of_ones=0; /* reset_number_of_ones*/
164
165
166
167
         return counter;
168
     }
169
     170
      * Gets FILE* f_in_ptr to read from encrypted text file and
171
      * FILE* f_out_ptr to write message to encoded file
172
      * return encrypted character number
173
174
175
     int
176
     decrypt_message(FILE *f_in_ptr, FILE *f_out_ptr)
177
     {
         /* Hint:While reading from encrypted text file check if character
178
                  equals to '*' or '_' and write to file 1 or 0
179
          *
180
          */
181
                         /* character, read from file */
         char number;
182
         int counter = 0; /*encrypted character number */
183
         /*end of local variables */
184
185
         while((fscanf(f_in_ptr,"%c",&number))!=EOF)/* read,end of file*/
186
187
188
             if(number!='\n')
             {
189
                 if(number=='*')
190
191
                 {
192
                     fprintf(f_out_ptr,"1");
193
                     counter++;
194
195
                 else if(number=='_')
196
                     fprintf(f_out_ptr,"0");
197
198
                     counter++;
199
                 }
200
             }
201
202
         return counter;
203
     }
204
     /* end of HW04 HASAN MEN 131044009 part2.c */
205
```