

```

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

```

```

75     /* exit program and print error if files couldn't be opened*/
76     if((f_plane_ptr= fopen(PLAINTEXTFILE,"w"))==NULL ||
77        (f_encoded_ptr= fopen(ENCODEDFILE,"r"))==NULL )
78     {
79         if((f_plane_ptr= fopen(PLAINTEXTFILE,"w"))==NULL)
80             printf("Can't open the PLAINTEXTFILE to write\n");
81         else
82             printf("Can't open the ENCODEDFILE to read.\n");
83     }
84
85     /*files opened , countinued */
86     else
87     {
88
89         /*call decode_message() */
90         decode_message(f_encoded_ptr, f_plane_ptr);
91         printf("\n**** ENCRYPTED ****\n");
92
93         /*close files*/
94         fclose(f_crypted_ptr);
95         fclose(f_encoded_ptr);
96     }
97 }
98 return 0;
99 /*end of main */
100 }
101
102 /*****
103  * Gets FILE* to write file and character to decode          *
104  * uses encoding table to convert encoded message to         *
105  * plain text message                                         *
106  *****/
107 void
108 decode_and_write_to_file(FILE *f_out_ptr, int number_of_ones)
109 {
110
111     /* write character according to number of ones */
112     switch(number_of_ones)
113     {
114         case 0: fprintf(f_out_ptr,"E"); break;
115         case 1: fprintf(f_out_ptr,"I"); break;
116         case 2: fprintf(f_out_ptr," "); break;
117         case 3: fprintf(f_out_ptr,"T"); break;
118         case 4: fprintf(f_out_ptr,"C"); break;
119         case 5: fprintf(f_out_ptr,"N"); break;
120         case 6: fprintf(f_out_ptr,"A"); break;
121         case 7: fprintf(f_out_ptr,"G"); break;
122         case 8: fprintf(f_out_ptr,"B"); break;
123         case 9: fprintf(f_out_ptr,"Z"); break;
124         case 10: fprintf(f_out_ptr,"H"); break;
125         case 11: fprintf(f_out_ptr,"L"); break;
126         case 12: fprintf(f_out_ptr,"U"); break;
127         case 13: fprintf(f_out_ptr,"V"); break;
128         case 14: fprintf(f_out_ptr,"R"); break;
129         case 15: fprintf(f_out_ptr,"S"); break;
130         case 16: fprintf(f_out_ptr,"Y"); break;
131     }
132 }
133
134 }
135
136 /*****
137  * Gets FILE* f_in_ptr to read from encoded text file and    *
138  * FILE* f_out_ptr to write message to plain text file        *
139  * return number of characters read from encoded text          *
140  *****/
141 int
142 decode_message(FILE *f_in_ptr, FILE *f_out_ptr)
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     int counter = 0;    /* number of 1-0 readed from file */
150     char character;     /* keeps 1-0 */
151     int number_of_ones=0; /* number of ones*/
152     /*end of local variables */
153
154     while((fscanf(f_in_ptr,"%c",&character))!=EOF)/*read until end of file*/
155     {
156         counter++;
157         if(character=='1')
158         {
159             number_of_ones++;
160         }
161         else if(character=='0')
162         {
163             decode_and_write_to_file(f_out_ptr,number_of_ones);
164             number_of_ones=0; /* reset number of ones*/
165         }
166     }
167     return counter;
168 }
169
170 /*****
171  * Gets FILE* f_in_ptr to read from encrypted text file and *
172  * FILE* f_out_ptr to write message to encoded file *
173  * return encrypted character number *
174  *****/
175 int
176 decrypt_message(FILE *f_in_ptr, FILE *f_out_ptr)
177 {
178     /* Hint:While reading from encrypted text file check if character
179     * equals to '*' or '_' and write to file 1 or 0
180     *
181     */
182     char number;    /* character, read from file */
183     int counter = 0; /*encrypted character number */
184     /*end of local variables */
185
186     while((fscanf(f_in_ptr,"%c",&number))!=EOF)/* read,end of file*/
187     {
188         if(number!='\n')
189         {
190             if(number=='*')
191             {
192                 fprintf(f_out_ptr,"1");
193                 counter++;
194             }
195             else if(number=='_')
196             {
197                 fprintf(f_out_ptr,"0");
198                 counter++;
199             }
200         }
201     }
202     return counter;
203 }
204
205 /* end of HW04_HASAN_MEN_131044009_part2.c */

```