```c
/*************************************************************
 *                                                           *
 * HW04 Q1                                                   *
 * Student Name: HASAN MEN                                   *
 * Student ID  : 131044009                                   *
 * Date        : 15.3.15                                     *
 * Points      : This program reads plain message from text  *
 * file and convert this 1/0 .It's not enough to share plain *
 * message because of cold war.Atfer that, takes 1/0 from    *
 * encode files and return that chars to "*","-" and "_" and *
 * write in the crypted message file.                        *
 *                                                           *
 *************************************************************/
#include <stdio.h>



#define PLAINTEXTFILE "Files/Q1/PlainMessagesToSent.txt"
#define ENCODEDFILE "Files/Q1/EncodedMessages.txt"
#define CRYPTEDFILE "Files/Q1/CryptedMessages.txt"

/*function prototypes */
/*************************************************************
 * Gets FILE* to write file and character to encode          *
 * uses encoding table to convert plain text to              *
 * encoded message                                           *
 *************************************************************/
void
encode_and_write_to_file(FILE *f_out_ptr, char character);

/*************************************************************
 * Gets FILE* f_in_ptr to read from plain text file and      *
 * FILE* f_out_ptr to write message to encoded file          *
 * return number of characters read from plain text          *
 *************************************************************/
int
encode_message(FILE *f_in_ptr, FILE *f_out_ptr);

/*************************************************************
 * Gets FILE* f_in_ptr to read from encoded text file and    *
 * FILE* f_out_ptr to write message to encrypted file        *
 * return encoded character number                           *
 *************************************************************/
int
crypt_message(FILE *f_in_ptr, FILE *f_out_ptr);

/*************************************************************
 * Reads plane text, creates encoded and crypted text files  *
 *************************************************************/
int
main(int argc, char* argv[])
{
    /* Start_of_main */

    FILE *f_plane_ptr, *f_encoded_ptr, *f_crypted_ptr;

    /* if files couldnot be opened , print error and exit the programs */
    if((f_plane_ptr= fopen(PLAINTEXTFILE,"r"))==NULL ||
       (f_encoded_ptr= fopen(ENCODEDFILE,"w"))==NULL )
    {
        if((f_plane_ptr= fopen(PLAINTEXTFILE,"r"))==NULL)
            printf("Can't open the PLAINTEXTFILE to read.\n");
        else
            printf("Can't open the ENCODEDFILE to write.\n");
    }

    /* files opened and continued */
    else
    {

    /* call encode_message and start to crypte message */
    encode_message(f_plane_ptr, f_encoded_ptr);

    /*close files*/
```

```
75      fclose(f_plane_ptr);
76      fclose(f_encoded_ptr);
77
78          /* if files couldnot be opened , print error and exit the programs */
79          if((f_crypted_ptr= fopen(CRYPTEDFILE,"w"))==NULL ||
80              (f_encoded_ptr= fopen(ENCODEDFILE,"r"))==NULL )
81          {
82              if((f_crypted_ptr= fopen(CRYPTEDFILE,"w"))==NULL)
83                  printf("Can't open the CRYPTEDFILE to write\n");
84              else
85                  printf("Can't open the ENCODEDFILE to read.\n");
86          }
87          /* files opened and continued */
88          else
89          {
90
91          /*call crypt_message funcion for last cryping*/
92          crypt_message(f_encoded_ptr, f_crypted_ptr);
93
94          /* Mission completed */
95          printf("\n****    CRYPTED   ****\n");
96
97          /*close files */
98          fclose(f_crypted_ptr);
99          fclose(f_encoded_ptr);
100         }
101     }
102     return 0;
103     /*end _of_main*/
104 }
105
106 /************************************************************
107  * Gets FILE* to write file and character to encode         *
108  * uses encoding table to convert plain text to             *
109  * encoded message                                          *
110  * Input:                                                   *
111  * ------characters which readed from plainfile            *
112  * Output:                                                  *
113  * ------encoded message(1/0's) -> writed in ENCODEDFILE    *
114  ************************************************************/
115 void
116 encode_and_write_to_file(FILE *f_out_ptr, char character)
117 {
118
119     int number_of_ones; /* numbers , readed from encoding table */
120     int ones;           /* local variable for loop */
121
122     /*  GTU Encoding Table */
123     switch(character)
124     {
125     case 'E':
126         number_of_ones=0;       break;
127     case 'I':
128         number_of_ones=1;       break;
129     case ' ':
130         number_of_ones=2;       break;
131     case 'T':
132         number_of_ones=3;       break;
133     case 'C':
134         number_of_ones=4;       break;
135     case 'N':
136         number_of_ones=5;       break;
137     case 'A':
138         number_of_ones=6;       break;
139     case 'G':
140         number_of_ones=7;       break;
141     case 'B':
142         number_of_ones=8;       break;
143     case 'Z':
144         number_of_ones=9;       break;
145     case 'H':
146         number_of_ones=10;      break;
147     case 'L':
148         number_of_ones=11;      break;
```

```
149      case 'U':
150          number_of_ones=12;      break;
151      case 'V':
152          number_of_ones=13;      break;
153      case 'R':
154          number_of_ones=14;      break;
155      case 'S':
156          number_of_ones=15;      break;
157      case 'Y':
158          number_of_ones=16;      break;
159
160      default : printf("*** %c NOT SUPPORTED  ***",character);
161      }
162
163      /* write to end of number of ones */
164      for(ones=1;ones<=number_of_ones;ones++)
165      {
166          fprintf(f_out_ptr,"1");
167      }
168      fprintf(f_out_ptr,"0");
169
170      /*end of funcion*/
171  }
172
173  /*************************************************************
174   * Gets FILE* f_in_ptr to read from plain text file and      *
175   * FILE* f_out_ptr to write message to encoded file          *
176   * return number of characters read from plain text          *
177   * Input:                                                    *
178   * ------PLAİNTEXTFİLE                                      *
179   * Output:                                                   *
180   * ------encoded message(1/0's) -> writed in ENCODEDFILE     *
181   *************************************************************/
182
183  int
184  encode_message(FILE *f_in_ptr, FILE *f_out_ptr)
185  {
186      char character; /* allocated to chars, readed files */
187      int counter = 0; /* number of chars read from input file */
188
189
190      while(fscanf(f_in_ptr,"%c",&character) != EOF)
191
192      {
193      /* s.times computer reads "\n" end of line and send this func. ,I made it
194       * to escape warnings."\n" not a character for us. */
195          if(character != '\n')
196          {
197
198          /* send chars to write file like 0/1's */
199          encode_and_write_to_file(f_out_ptr,character);
200          counter++;  /* increase number of read */
201          }
202
203          /* if reads '\n' , scan next character to come end of file*/
204          else fscanf(f_in_ptr,"%c",&character);
205      }
206      return counter;
207  }
208
209  /*************************************************************
210   * Gets FILE* f_in_ptr to read from encoded text file and    *
211   * FILE* f_out_ptr to write message to encrypted file        *
212   * return number of characters read from encoded text file   *
213   *************************************************************/
214  int
215  crypt_message(FILE *f_in_ptr, FILE *f_out_ptr)
216  {
217      char number;    /* local variable, keep 1/0 from file */
218      int counter = 0;    /* */
219      int read_number=0; /* number of characters read file*/
220      int N=5;
221
222      while(fscanf(f_in_ptr,"%c",&number) != EOF)
```

```
223       {
224           int M=N;
225   
226       /* reading char from file if it's 1 prints '*' , if it's 0 print '0'*/
227       /* and increase read number and inrease number                       */
228       /* if counter==m print '-' and reduce N-- when N==0 -> make N==5     */
229       /* *****-****-***-**-*-*****-****-***-**-*                           */
230       /* make a loof like up there                                         */
231           if(number!='\n')
232           {
233   
234               if(number=='1')
235                   fprintf(f_out_ptr,"*");
236               else fprintf(f_out_ptr,"_");
237               counter++;
238               read_number++;
239           }
240   
241           if(counter==M)
242           {
243               fprintf(f_out_ptr,"-");
244               counter=0;
245               N--;
246   
247               if(N==0)
248               {
249                   N=5;
250               }
251           }
252       }
253       return read_number;
254   }
255   
256   /* END OF HW04_HASAN_MEN_131044009_part1.c   */
```