```c
/**********************************************************
 *                                                        *
 * HW04 Q1                                                *
 * Student Name: MUSTAFA AKILLI                           *
 * Student ID  : 131044017                                *
 * Date        : 15 march 2015                            *
 * Points      : 29                                       *
 *                                                        *
 **********************************************************/
#include <stdio.h>

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

void encode_and_write_to_file(FILE *f_out_ptr, char character);
int encode_message(FILE *f_in_ptr, FILE *f_out_ptr);
int crypt_message(FILE *f_in_ptr, FILE *f_out_ptr);

int main(int argc, char* argv[])
{
    int counter,counter2;

    FILE *f_plane_ptr, *f_encoded_ptr, *f_crypted_ptr;

    f_plane_ptr = fopen(PLAINTEXTFILE,"r");
    f_encoded_ptr = fopen(ENCODEDFILE,"w");

    if(f_plane_ptr == NULL){

        printf("ERROR!! Plain text file could not be opened to read.\n");
        return 0;
    }

    if(f_encoded_ptr == NULL){

        printf("ERROR!! Encoded text file could not be opened to write.\n");
        return 0;
    }

    counter = encode_message(f_plane_ptr, f_encoded_ptr);

    fclose(f_encoded_ptr);
    fclose(f_plane_ptr);

    f_encoded_ptr = fopen(ENCODEDFILE,"r");
    f_crypted_ptr = fopen(CRYPTEDFILE,"w");

    if(f_encoded_ptr == NULL){

        printf("ERROR!! Plain text file could not be opened to read.\n");
        return 0;
    }

    if(f_crypted_ptr == NULL){

        printf("ERROR!! Encoded text file could not be opened to write.\n");
        return 0;
    }

    counter2 = crypt_message(f_encoded_ptr, f_crypted_ptr);

    fclose(f_encoded_ptr);
    fclose(f_crypted_ptr);

    return 0;
}


/**********************************************************
 * 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)
{
    int counter=1;

    switch(character){
        case 'E' : while(counter<1){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'I' : while(counter<2){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case ' ' : while(counter<3){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'T' : while(counter<4){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'C' : while(counter<5){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'N' : while(counter<6){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'A' : while(counter<7){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'G' : while(counter<8){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'B' : while(counter<9){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'Z' : while(counter<10){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
                   break;
        case 'H' : while(counter<11){
                   fprintf(f_out_ptr,"1");
                   ++counter;
                   }
                   fprintf(f_out_ptr,"0");
```

```c
                      break;
        case 'L' : while(counter<12){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
        case 'U' : while(counter<13){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
        case 'V' : while(counter<14){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
        case 'R' : while(counter<15){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
        case 'S' : while(counter<16){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
        case 'Y' : while(counter<17){
                      fprintf(f_out_ptr,"1");
                      ++counter;
                      }
                      fprintf(f_out_ptr,"0");
                      break;
    }
}

/**************************************************************
 * 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)
{

    char status;
    char character;
    int counter = 0;

    status = fscanf(f_in_ptr,"%c",&character);

    while(status != EOF){

        encode_and_write_to_file(f_out_ptr,character);
        status = fscanf(f_in_ptr,"%c",&character);
        ++counter;
    }

    return counter;
}

/**************************************************************
 * Gets FILE* f_in_ptr to read from encoded text file and    *
 * FILE* f_out_ptr to write message to encrypted file        *
 * return number of characters read from encoded text file   *
 **************************************************************/

int crypt_message(FILE *f_in_ptr, FILE *f_out_ptr)
{
    int counter = 0;
    int encoded;
    int status;
    int M=0;
    int N=5;
    int temp=5;

    status = fscanf(f_in_ptr,"%1d",&encoded);

    while(status != EOF){

        if(encoded==1){
            fprintf(f_out_ptr,"*");
        }

        else{
            fprintf(f_out_ptr,"_");
        }

        --N;

        if(N==M){
            fprintf(f_out_ptr,"-");
            --temp;

            if(temp>N){
                N = temp;
            }
            else{
                temp=5;
                N=5;
            }
        }

        status = fscanf(f_in_ptr,"%1d",&encoded);
        ++counter;
    }

    return counter;
}
```