```c
/************************************************************
 *                                                          *
 * HW04 Q3                                                  *
 * Student Name: HASAN MEN                                  *
 * Student ID  : 131044009                                  *
 * Date        : 15.3.15                                    *
 * Points      : this program decodes X University's        *
 *   messaging system! Assume X University uses a heuristic *
 *  to create encoding table. Their heuristic is just based *
 *  on frequency of letters. Frequently used letters have   *
 *  shorter code length.                                    *
 *                                                          *
 ************************************************************/
#include <stdio.h>

#define TRUE 1
#define FALSE 0
#define CHARACTERFILE "Files/Q3/CharacterList.txt"
#define SAMPLEFILE "Files/Q3/Sample.txt"
#define ENCODEDFILE "Files/Q3/XUniversityEncoded.txt"
#define PLAINTEXTFILE "Files/Q3/XUniversityMessage.txt"

/************************************************************
 * Swaps values of two integers                             *
 ************************************************************/
void
swap_int(int *a, int *b);

/************************************************************
 * Swaps values of two characters                           *
 ************************************************************/
void
swap_char(char *a, char *b);

/************************************************************
 * Sorts characters according to counts. At the end         *
 * make sure that *c1 keeps most frequent used letter, *c3   *
 * keeps least frequent used letter and *c2 keeps remained   *
 * letter                                                   *
 ************************************************************/
void
sort(char *a, int a_num, char *b, int b_num, char *c, int c_num);

/************************************************************
 * Check whether character is big ASCII letter or not        *
 * return TRUE or FALSE                                     *
 ************************************************************/
int
is_letter(char c);

/************************************************************
 * Read characters from character list file and if character *
 * is letter assign characters to c1, c2 and c3.            *
 * If file has not three letters assign NULL to input char   *
 * by order. For ex. file has two letters assign proper      *
 * letters to c1 and c2 and assign NULL to c3. If file has   *
 * four letters assign c1, c2 and c3 first three letters.    *
 * Return number of letters in character list file.         *
 * Do not forget to count only proper letters with your      *
 * is_letter function. Return number of letters not chars    *
 ************************************************************/
int
read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3);

/************************************************************
 * Read letters from Sample file and compute frequency of    *
 * letters. Then sort it inside this function. Call sort     *
 * function. At the end make sure that *c1 keeps most        *
 * frequent used letter, *c3 keeps least frequent used       *
 * letter and *c2 keeps remained letter                     *
 ************************************************************/
void
count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3);

```

```c
/***********************************************************
 * Read from XUniversityEncoded file to decode message and   *
 * write decoded (plain text) message to XUniversityMessage  *
 * file. Make sure c1 keeps most frequent used letter,  c3   *
 * keeps least frequent used letter and  c2 keeps remained   *
 * letter while calling function. According to frequency     *
 * you know their codes. c1: 0, c2: 10, c3: 110.             *
 ***********************************************************/
void
decode(FILE *f_in_ptr, FILE *f_out_ptr, char c1, char c2, char c3);

/***********************************************************
 * Learns XUniversity's encoding system from given files ,   *
 * decodes their encoded messages and writes as plain text to*
 * a file                                                    *
 ***********************************************************/
int
main(int argc, char* argv[])
{
    FILE *f_character_list_ptr, *f_sample_file_ptr, *f_encoded_ptr,
        *f_plain_text_ptr;
    char c1, c2, c3;

    /* exit progtam and print error if files couldn't be opened*/
    if((f_character_list_ptr=fopen(CHARACTERFILE,"r"))==NULL)
    {
        printf("\nCan't open CHARACTERFILE to read\n");
    }
    else
    {

        /* call read_character_list and assign chars to c1,c2,c3 */
        /* continue if c1,c2,c3 not null */
        if(read_character_list(f_character_list_ptr, &c1, &c2, &c3)==3)
        {

            /* close c.list file */
            fclose(f_character_list_ptr);



            /* exit progtam and print error if files couldn't be opened*/
            if((f_sample_file_ptr=fopen(SAMPLEFILE,"r"))==NULL)
            {
                printf("\nCan't open SAMPLEFILE to read\n");
            }
            else
            {

                /* call count letter func. */
                count_letters(f_sample_file_ptr,&c1,&c2,&c3);

                /* close sample file*/
                fclose(f_sample_file_ptr);

                /* exit progtam and print error if files couldn't be opened*/
                if((f_encoded_ptr=fopen(ENCODEDFILE,"r"))==NULL ||
                    (f_plain_text_ptr=fopen(PLAINTEXTFILE,"w"))==NULL )
                {
                    if((f_encoded_ptr=fopen(ENCODEDFILE,"r"))==NULL)
                        printf("Can't open ENCODEDFILE to read");
                    else printf("Can't open PLAINTEXTFILE to write");
                }

                /*files opened */
                else
                {

                /* call decode function and write new message */
                decode(f_encoded_ptr,f_plain_text_ptr,c1,c2,c3);

                /* close files */
                fclose(f_encoded_ptr);
                fclose(f_plain_text_ptr);
                }
            }
        }
        else printf("\nNumber of letter not equal 3 - Program finished\n");
    }

    printf("****    Mission Completed   ****\n");

    return 0;
}

/************************************************************
 * Swaps values of two integers                             *
 ************************************************************/
void
swap_int(int *a, int *b)
{
    int term;
    term = *a;
    *a=*b;
    *b=term;
}

/************************************************************
 * Swaps values of two characters                           *
 ************************************************************/
void
swap_char(char *a, char *b)
{
    char term;
    term = *a;
    *a=*b;
    *b=term;
}

/************************************************************
 * Sorts characters according to counts. At the end          *
 * make sure that *c1 keeps most frequent used letter, *c3   *
 * keeps least frequent used letter and *c2 keeps remained   *
 * letter                                                    *
 ************************************************************/
void
sort(char *a, int a_num, char *b, int b_num, char *c, int c_num)
{

    /* sort c1,c2,c3 accoring to most frequent used */
    if(a_num<=b_num)
    {
        swap_char(a,b);
        swap_int(&a_num,&b_num);
    }
    if(a_num<=c_num)
    {
        swap_char(a,c);
        swap_int(&a_num,&c_num);
    }

    if(b_num<=c_num)
    {
        swap_char(b,c);
        swap_int(&b_num,&c_num);
    }

    /* printf("%c=%d %c=%d %c=%d\n",*a,a_num,*b,b_num,*c,c_num); */
    /* check c1,c2,c3 and thehir used numbers after shorting*/
}

/************************************************************
 * Check whether character is big ASCII letter or not        *
 * return TRUE or FALSE                                      *
 ************************************************************/
int
is_letter(char c)
{
    if(c<='Z' && c >='A')
        return TRUE;
    else
        return FALSE;
}

/**********************************************************
 * Read characters from character list file and if character *
 * is letter assign characters to c1, c2 and c3.             *
 * If file has not three letters assign NULL to input char   *
 * by order. For ex. file has two letters assign proper      *
 * letters to c1 and c2 and assign NULL to c3. If file has   *
 * four letters assign c1, c2 and c3 first three letters.    *
 * Return number of letters in character list file.          *
 * Do not forget to count only proper letters with your      *
 * is_letter function. Return number of letters not chars    *
 ***********************************************************/
int
read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3)
{
    int counter = 0;    /*local variable, counts proper letter */
    char character;  /* keeps character from file*/
    while((fscanf(f_in_ptr," %c",&character))!=EOF)
    {
        if(is_letter(character)==TRUE)
        {
            counter++;

            if(counter==1)
            {
                *c1=character;
                *c2=0; /* c2=null*/
            }
            else if(counter==2)
            {
                *c2=character;
                *c3=0; /* c3=null*/
            }
            else if(counter==3)
            {
                *c3=character;
            }
        }
    }
    return counter;
}

/**********************************************************
 * Read letters from Sample file and compute frequency of    *
 * letters. Then sort it inside this function. Call sort     *
 * function. At the end make sure that *c1 keeps most        *
 * frequent used letter, *c3 keeps least frequent used       *
 * letter and *c2 keeps remained letter                      *
 ***********************************************************/
void
count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3)
{
    char letter; /* keeps character from sample file*/
    int a_num=0,b_num=0,c_num=0;

    while(fscanf(f_in_ptr," %c",&letter)!=EOF)
    {
        if(letter==*c1)
        {
            a_num++;
        }
        else if(letter==*c2)
        {
            b_num++;
        }
        else if(letter==*c3)
        {
            c_num++;
        }
    }
    /*check c1,c2,c3 and their number before shorting */
    /*printf("%c=%d %c=%d %c=%d\n",*c1,a_num,*c2,b_num,*c3,c_num);*/
    sort(c1,a_num,c2,b_num,c3,c_num);
}

/**************************************************************
 * Read from XUniversityEncoded file to decode message and    *
 * write decoded (plain text) message to XUniversityMessage   *
 * file. Make sure c1 keeps most frequent used letter,  c3    *
 * keeps least frequent used letter and  c2 keeps remained    *
 * letter while calling function. According to frequency      *
 * you know their codes. c1: 0, c2: 10, c3: 110.              *
 **************************************************************/
void
decode(FILE *f_in_ptr, FILE *f_out_ptr, char c1, char c2, char c3)
{
    char character;
    int number_of_ones=0;
    while(fscanf(f_in_ptr," %c",&character)!=EOF)
    {

        /* count number of  ones until take '0' and write c1,c2,c3 according
         * known codes */
        if(character=='1')
        {
            number_of_ones++;
        }
        else if(character=='0')
        {
            switch(number_of_ones)
            {
                case 0: fprintf(f_out_ptr,"%c",c1); break;
                case 1: fprintf(f_out_ptr,"%c",c2); break;
                case 2: fprintf(f_out_ptr,"%c",c3); break;
            }
            number_of_ones=0; /* rest the number of ones */
        }
    }
}

/* end of HW04_HASAN_MEN_131044009_part3.c */
```