```c
/*###########################################################################*/
/*HW04_Furkan_Erdol_131044065_part3.c                                        */
/*______________________________                                             */
/*Written by Furkan Erdol on March 12, 2015                                  */
/*Description                                                                */
/*____________                                                               */
/*                                                                           */
/*>>>>Learns XUniversity's encoding system from given files, decodes their   */
/*encoded messages and writes as plain text to a file<<<<                    */
/*                                                                           */
/*Inputs:                                                                    */
/*  -Character list                                                          */
/*  -Sample                                                                  */
/*  -X University encoded message                                            */
/*Outputs:                                                                   */
/*  -X University message                                                    */
/*                                                                           */
/*Points: 49                                                                 */
/*...........................................................................*/
/*                              Includes                                     */
/*...........................................................................*/
#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.                                    *
 ****************************************************************************/
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 example 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*
 ****************************************************************************/
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);

/****************************************************************************
 * 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);



int
main(void)
{

    int character_number; /* Number of characters read */
    char c1, c2, c3;      /* Letters of messaging system’s alphabet */

    FILE *f_character_list_ptr, /* CharacterList.txt */
         *f_sample_file_ptr,    /* Sample.txt */
         *f_encoded_ptr,        /* XUniversityEncoded.txt */
         *f_plain_text_ptr;     /* XUniversityMessage.txt */


    /* Open character list file */
    f_character_list_ptr=fopen(CHARACTERFILE, "r");

    /*Exit program and print error if character list file could not be opened to read */
    if(f_character_list_ptr==NULL)
    {
        printf("CharacterList.txt couldn't open...\n");
        return 0;
    }

    /* Read letters of messaging system’s alphabet and get return value of number of letter */
    character_number=read_character_list(f_character_list_ptr, &c1, &c2, &c3);

    /*if number of letter read is not equal to three exit program */
    if(character_number!=3)
    {
        printf("Warning: Number of letter not equal to three...\n");
        return 0;
    }

    /* Close character list */
    fclose(f_character_list_ptr);

    /* Open sample file */
    f_sample_file_ptr=fopen(SAMPLEFILE, "r");

    /* Exit program and print error if sample file could not be opened to read */
    if(f_sample_file_ptr==NULL)
    {
        printf("Sample.txt couldn't open...\n");
        return 0;
    }

    /* Read letters from Sample file and compute frequency of letters. Then   *
     * sort it inside this function                                           */
    count_letters(f_sample_file_ptr, &c1, &c2, &c3);

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

    /* Open encoded file and plain text file */
    f_encoded_ptr=fopen(ENCODEDFILE, "r");
    f_plain_text_ptr=fopen(PLAINTEXTFILE, "w");

    /*Exit program and print error if encoded file could not be opened to read */
    if(f_encoded_ptr==NULL)
    {
        printf("XUniversityEncoded.txt couldn't open...\n");
        return 0;
    }

    /*Exit program and print error if plain text file could not be opened to write */
    if(f_plain_text_ptr==NULL)
    {
        printf("XUniversityMessage.txt couldn't open...\n");
        return 0;
    }

    /* Read from XUniversityEncoded file to decode message and write decoded  *
     * (plain text) message to XUniversityMessage file                        */
    decode(f_encoded_ptr, f_plain_text_ptr, c1, c2, c3);

    /* Close encoded file and plain text file */
    fclose(f_encoded_ptr);
    fclose(f_plain_text_ptr);

    return 0;


}


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

    int temp;

    temp=*a;

    *a=*b;
    *b=temp;

}

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

    char temp; /* Temporary variable */

    temp=*a;

    *a=*b;
    *b=temp;
}

/******************************************************************************
 * Sorts characters according to counts.                                      *
 ******************************************************************************/
void sort(char *a, int a_num, char *b, int b_num, char *c, int c_num)
{

    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);
    }
}

/******************************************************************************
```

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

    if(c>=(int)'A'&&c<=(int)'Z')
        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 example 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*
 *****************************************************************************/
int read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3)
{
    char character; /* Read character variable */

    int counter = 0; /* Number of letters in file */


    while(fscanf(f_in_ptr, "%c", &character)!=EOF) /* Reads the file until the end */
    {
        if(is_letter(character)==TRUE) /* Call is_letter function */
        {
            counter++;

            if(counter==1) /* The first letter is assigned to c1 */
                *c1=character;
            else if(counter==2) /* The second letter is assigned to c2 */
                *c2=character;
            else if(counter==3) /* The third letter is assigned to c3 */
                *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)
{
    int count_c1=0, /* Counts frequency of letter c1 */
        count_c2=0, /* Counts frequency of letter c2 */
        count_c3=0; /* Counts frequency of letter c3 */

    char character; /* Read character variable */

    while(fscanf(f_in_ptr, "%c", &character)!=EOF) /* Reads the file until the end */
    {
        if(character==*c1)
            count_c1++;
        else if(character==*c2)
            count_c2++;
        else if(character==*c3)
            count_c3++;
    }

    sort(c1, count_c1, c2, count_c2, c3, count_c3); /* Call short function */
}
```

```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)
{
    char character; /* Read character variable */

    int count_number_of_1=0; /* Counts number of 1 */


    while(fscanf(f_in_ptr, "%c", &character)!=EOF) /* Reads the file until the end */
    {

        if(character=='1')
        count_number_of_1++;
        else
        {

            switch(count_number_of_1){

            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;
            }

        count_number_of_1=0;

        }

    }

}

/*###########################################################################*/
/*                End of HW04_Furkan_Erdol_131044065_part3.c                 */
/*###########################################################################*/
```