```c
/*###################################################################*/
/*                                                                   */
/*  HW06_PART2                                                       */
/* Tarih : 6.4.15                                                    */
/* Hazirlayan : HASAN MEN                                            */
/*                                                                   */
/*Dosyadan okunan kelimelerin buyuk ve kucuk unlu uyumuna            */
/*uygunluklarini bulup kelimelerin cogul hallerini yeni dosyaya      */
/*yazdiran program parcacigi                                         */
/*###################################################################*/
#include <stdio.h>
#include <string.h>

/* dosya isimleri - constant macro olarak */
#define VOWEL "Vowels.txt"
#define NOUN "Nouns.txt"
#define PLURAL "Plural.txt"

#define SIZE 20     /* maximum string boyutu */
#define STR_LEN 4   /* kac satirlik okuma yapilacak */

typedef enum{HARD,SOFT,CONS_MAJ} major_type;    /* buyuk unlu uyumu */
typedef enum{FLAT,ROUND,CONS_MIN} minor_type;   /* kucuk unlu uyumu */
typedef enum{FALSE,TRUE} bool;
/* enumerated type definitions */


/* foksiyon prototipleri */

/* gonderilen gelimenin buyuk unlu uyumuna uygunlugu kontrol edilir*/
bool is_major_vh_word(const char* word, const char* v_hard, const char* v_soft);

/* buyuk unlu uyumu icin harflerin kalin yada yumussaklik durumunu dondurur*/
major_type major(const char ch1, const char* v_hard, const char* v_soft);

/* kucuk unlu uyumuna uygunluk kontrolu yapilir */
bool is_minor_vh_word( const char* word, const char* v_flat, const char* v_round);

/*kucuk unlu uyumu icin yuvarlak ve duzluk kontrol edilir */
minor_type minor(const char ch1, const char* v_flat, const char* v_round);

/* kelimeyi cogul yapmak icin son unlu harfin turune bakilir */
major_type find_last_type(const char* word,const char* v_hard, const char* v_soft);


/* find_last_type kullanarak son sesli harfe gore kelimeleri cogullastirir*/
char* make_plural(const char* noun , char* plural_noun,const char* v_hard, const char* v_soft );


int main()
{

    char line[10];
    char noun[SIZE][SIZE];
    char plural[SIZE][SIZE];

    char hard[STR_LEN];
    char soft[STR_LEN];
    char flat[STR_LEN];
    char round[STR_LEN];
    int num_noun=0;

    int i=0,j;
    char status;

    bool major,minor;


    FILE *vp=fopen(VOWEL,"r");
    FILE *np=fopen(NOUN,"r");
    FILE *pp=fopen(PLURAL,"w");
    /* degiskenlerimiz */

    if(vp==NULL || np==NULL)    /* dosyalarin acilip acilmama kontrolu */
        printf("Files couldn't opened to read !!!");
    else
    {
        printf("Reading the vowels..\n");
        while(fgets(line,SIZE,vp)!=NULL)
        {
            /* EOF a kadar satir satir oku ve satirlari sirayla hard soft
            flat ve rounda ata*/

            if(line[strlen(line)-1]=='\n')
                line[strlen(line)-1]='\0';

            switch(i)
            {
                case 0: strcpy(hard,line); break;
                case 1: strcpy(soft,line); break;
                case 2: strcpy(flat,line); break;
                case 3: strcpy(round,line); break;
            }
            i++;
        }

        printf("Reading the nouns..\n");
        while(fscanf(np,"%s",noun[num_noun])!=EOF) /* dosyadaki kelimleri okur*/
            num_noun++;

        printf("The maintaince of the harmonies\n");
        printf("%8c%8s%2c%5s\n",' ',"MAJOR",' ',"MINOR");
        /* tum kelimelerin tek tek uyumluluklarinin bulunmasi ve cogullastirma*/
        for(j=0;j<num_noun;j++)
        {

            printf("%-10s",noun[j]);/* kelimeyi ekrana basalim */
            major = is_major_vh_word(noun[j],hard,soft); /* major kontrol */
            if(major)   /* durumlari ekrana bas */
                printf("%3cT",' ');
            else printf("%3cF",' ');

            minor=is_minor_vh_word(noun[j],flat,round); /* minor kontrol */
            if(minor)   /* durumlari ekrana bas */
                printf("%3cT",' ');
            else printf("%3cF",' ');
            printf("\n");


            make_plural(noun[j],plural[j],hard,soft);   /* cogullastirma */
            fprintf(pp,"%s ",plural[j]);
        }
        printf("Wrote the plurals of the nouns!!!\n");


        fclose(vp);
        fclose(np);
        fclose(pp);
        /* dosyalarin kapanmasi */
    }
    return 0;
}
bool is_major_vh_word(const char* word, const char* v_hard, const char* v_soft)
{
    int i;
    int hardd=0;    /* kalin unlu sayisi */
    int softt=0;    /* yumusak unlu sayisi*/
    int cons=0; /* unsuz sayisi */
    major_type mjr; /* major kontrolu ucun enumerated type */

    for(i=0;i<(int)strlen(word);i++)/* wordp aracalara ayirarar harf harf bak*/
    {
            mjr = major(word[i],v_hard,v_soft); /* harfin geri donus degeri */
            if(mjr==HARD)
```

```
                hardd++;
            else if(mjr==SOFT)
                softt++;
            else cons++;
    }

    if(hardd!=0 && softt==0)    /* soft harf yoksa buyuk unluye uyar */
        return TRUE;
    else if(hardd==0 && softt!=0)   /* hard harf yoksa buyuk unlu uyar */
        return TRUE;
    else return FALSE;  /* ikiside varsa uymaz */



}

major_type major(const char ch1, const char* v_hard, const char* v_soft)
{

    int i;

    /* gelen harf hard stringinin icindekilerle karsilastir */
    for(i=0;i<(int)strlen(v_hard);i++)
    {
        if(ch1==v_hard[i])
            return HARD;
    }

    /* gelen harf soft stringinin icindekilerle karsilastir */
    for(i=0;i<(int)strlen(v_soft);i++)
    {
        if(ch1==v_soft[i])
            return SOFT;
    }

    return CONS_MAJ;    /* esitlik bulunmazsa harf unsuzdur */

}

bool is_minor_vh_word( const char* word, const char* v_flat, const char* v_round)
{

    int i;
    int flatt=0;    /* duz sayisi */
    int roundd=0;   /* yuvarlak sayisi */
    int cons=0; /* unsuz sayisi */
    major_type min;

    /* kelimeleri tek tek  parcalayarak inceleme */
    for(i=0;i<(int)strlen(word);i++)
    {
            min = major(word[i],v_flat,v_round); /* harfin donus degeri */
            if(min==HARD)
                flatt++;
            else if(min==SOFT)
                roundd++;
            else cons++;
    }

    if(flatt!=0 && roundd==0)   /* round harf yoksa kucuk unluye uyar */
        return TRUE;
    else if(flatt==0 && roundd!=0)  /* flatt harf yoksa kucuk unluye uyar */
        return TRUE;
    else return FALSE;  /* ikiside varsa yada yoksa kucuk unluye uymaz */

}

minor_type minor(const char ch1, const char* v_flat, const char* v_round)
{

    int i;

```

```c
    for(i=0;i<(int)strlen(v_flat);i++)  /* flat stringinden tek tek kontrol */
    {
        if(ch1==v_flat[i])
            return FLAT;
    }
    for(i=0;i<(int)strlen(v_round);i++) /* round stringinden tek tek kontrol */
    {
        if(ch1==v_round[i])
            return ROUND;
    }

    return CONS_MIN;    /* flat round yoksa consotant return eder */

}

/* son unlunun hard yada softlugunu kontrol eder */
major_type find_last_type(const char* word,const char* v_hard, const char* v_soft)
{
    int i;
    major_type first;
    major_type last;

    for(i=0;i<(int)strlen(word);i++)
    {
        first=major(word[i],v_hard,v_soft); /* major tipine bakildi */
        if(first==HARD || first==SOFT) /* sessiz degilse tipi donduruldu */
            last=first; /* unlu degerimiz  buraya assign edildi */

    }
    return last;    /* enumerated type olarak return edildi */

}
char* make_plural(const char* noun , char* plural_noun ,const char* v_hard, const char* v_soft)
{
    int i;

    major_type last= find_last_type(noun,v_hard,v_soft); /* son unlunun donusu*/
    strcpy(plural_noun,noun);   /* plural_noun stringine kelimemiz yazilir*/

    if(last==HARD)  /* hard yada softluga gore eklerimiz plurala eklenir */
        strcat(plural_noun,"lar");
    else if(last==SOFT)
        strcat(plural_noun,"ler");

}

/* HW06_HASAN_MEN_131044009_part2.c SONU */
```