```
/*########################################################################*/
/*  HW08_HASAN_MEN_131044009_part2.c                                      */
/*  HAZIRLAYAN  : HASAN MEN - 131044009                                   */
/*  TARIH       : 20.4.15                                                 */
/*                                                                        */
/*  TANIM       : Dosyadan randevu zamanlari ve hasta bilgilerini alip    */
/*  uygun tc lere isimleri eslestiren , eş zamanli randevulardan          */
/*  sonuncusunu silip ekrana degisim bilgilerini ve dosyaya randevulari   */
/*  artan sure siralamasina gore yazan program parcacigi :)               */
/*########################################################################*/

#include<stdio.h>
#include<string.h>

#define MAX_SZ 100  /* Aralarda tanimlanan arraylar icin max boyutlar*/

typedef enum    /* cinsiyet yapisi */
{
    MALE,FEMALE
}Gender_t;

typedef enum    /* aylar */
{
    JANUARY=1,FEBRUARY,MARCH,APRIL,MAY,JUNE,
    JULY,AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
}Months_t;

typedef struct  /*tc numarasi */
{
    int first_half; /*6digit*/
    int second_half; /*5digit*/

}TCid_no_t;

typedef struct  /* kisiler */
{
    TCid_no_t id_no;
    char name[30];
    char surname[30];
    Gender_t gender;
}People_t;

typedef struct  /* zaman */
{
    int hour;
    int minute;
}Time_t;

typedef struct /* randevu-zaman bilgileri */
{
    int year;
    Months_t month;
    int day;
    Time_t time;
}Date_t;

typedef struct  /* randevu structi*/
{
    People_t people;
    Date_t date;
}Appointment_t;

/* Dosyadan kisi isimlerini, tc lerini ve cinsiyetlerini okur */
int get_people(const char *file_name,People_t people[],int max_size);

/* Dosyadan tcleri ve randevu zamanlari okur */
int get_appointments(   const char *file_name,
                        Appointment_t appointments[],int max_size);
/*isimve tc leri olan kisilere, tclerine gore rand .zamanlarini assign eder*/
void write_names(   Appointment_t Appointment_t[],
                    int size_app,const People_t people[],int size_people);
```

```
/* randevu zamanlarina gore kisileri siralar (kucukten buyuge)*/
int sort_appointments(Appointment_t appointments[],int size);

/* es zamanli randevulardan bazilarini siler */
int check_appointments(Appointment_t appointments[],int size);

/* duzenlenmis listelerin output dosyasina bazilamasi */
void write_appointments(    const char *file_name,
                            Appointment_t appoinments[],int size);

int main()
{

    People_t human[MAX_SZ]; /* kisilerin okunacagi struct array */
    Appointment_t appoint[MAX_SZ];  /* randevularin okunacagi yer */
    int rec_read_people,
        rec_read_appoin;   /*okunan kisi ve randevu sayilari*/
    /* input-output  dosyala isimleri */
    char file_people[]="People.txt";
    char file_app_req[]="AppointmentReqs.txt";
    char file_appoin[]="Appointments.txt";
    int new_app_size;
    /* degiskenlerin sonu */



    /*kisilerin ve randavularin okunmasi */
    rec_read_people = get_people(file_people,human,MAX_SZ);
    rec_read_appoin = get_appointments(file_app_req,appoint,MAX_SZ);

    /* okunan bilgilerin ekrana basilmasi */
    if(rec_read_people)
    {
        printf("->%d People Records founded...\n",rec_read_people);

        if(rec_read_appoin)
        {
            printf("->%d Appoinments Record founded...\n",rec_read_appoin);

            /* kisi-tc eslestirmesi yapilir */
            write_names(appoint,rec_read_appoin,human,rec_read_people);
            /*ayni zamanli randevular silindi */
            new_app_size=check_appointments(appoint,rec_read_appoin);
            printf("->%d appointments rejected...\n",
                                                rec_read_appoin-new_app_size);
            /* zamana gore siralandi */
            sort_appointments(appoint,new_app_size);
            /*out dosyasina bilgiler yazildi */
            write_appointments(file_appoin,appoint,new_app_size);
            printf("->%d Appoinments saved...\n",new_app_size);
        }
        else printf("File couldn't opened to  read appoinment reacords");
    }
    else printf("File couldn't opened to  read people reacords");

    return 0;
}

int get_people(const char *file_name,People_t people[],int max_size)
{

    int status;
    int i=0; /*Sayac*/
    int num_records=0;
    char genderr;   /* cinsiyet aktarimi icin */
    FILE *peopleP;
    peopleP = fopen(file_name,"r");

    /* dosya acilmaz ise hata mesaji dondur */
    if(peopleP==NULL)
        num_records=0;
    else
```

```
    {
        do
        {
        status = fscanf(peopleP,"%6d%5d",   &(people[i].id_no.first_half),
                                            &(people[i].id_no.second_half));
        status = fscanf(peopleP,"%s %s",    people[i].name,people[i].surname);
        status = fscanf(peopleP," %c",      &genderr);
        people[i].gender=(Gender_t)genderr;
        i++;
        num_records=i-1;

        }while(status!=EOF);
        /*eof olana kadar veriler okundu*/
    }

    fclose(peopleP);
    /* okunan kisi sayisi donduruldu*/
    return num_records;
}

int get_appointments(   const char *file_name,
                        Appointment_t appointments[],int max_size)
{
    int status;
    int i=0; /*Sayac*/
    int num_rec_appoin=0;
    char twodot;    /* saat dilimindeki ':'ayrimdan kacmak icin*/
    int monthh; /*ay donusumu yapilacak */
    FILE *appoinP;
    appoinP = fopen(file_name,"r");

    /* okuna yapilmazsa hata ver */
    if(appoinP==NULL)
        num_rec_appoin=0;
    else
    {

        do{
       status = fscanf(appoinP,"%6d%5d%d%d%d%d %c%d",
                            &(appointments[i].people.id_no.first_half),
                            &(appointments[i].people.id_no.second_half),
                            &(appointments[i].date.year),
                            &monthh,
                            &(appointments[i].date.day),
                            &(appointments[i].date.time.hour),
                            &twodot,
                            &(appointments[i].date.time.minute));
        appointments[i].date.month=(Months_t)monthh;
        i++;
        num_rec_appoin=i-1;

        }while(status!=EOF);
        /* eof gorene kadar verileri oku*/
    }

    fclose(appoinP);
    /* okunan randevu sayisi return edildi */
    return num_rec_appoin;
}

void write_names(   Appointment_t appointments[],int size_app,
                    const People_t people[],int size_people)
{
    int i=0,j=0;

    /*eslesen tclere isimleri yerlestirir. */
    for(i=0;i<size_people;i++)
        for(j=0;j<size_app;j++)
            if(people[i].id_no.first_half==appointments[j].people.id_no.first_half &&
               people[i].id_no.second_half==appointments[j].people.id_no.second_half )
            {
                strcpy(appointments[i].people.name,people[i].name);
```

```
                strcpy(appointments[i].people.surname,people[i].surname);
                appointments[i].people.gender=people[i].gender;
            }
}

/* bobble sort */
int sort_appointments(Appointment_t appointments[],int size)
{
    /* zamana gore siralama yapildi - kucukten buyuge */
    int i,j;
    Appointment_t temp; /* tur cakismamasi icin temp  */

    for(i=0;i<size;i++)
    {
        for(j=0;j<size-1;j++)
        {
            /* printf("i=%d-j=%d\n",i,j); */
            if(appointments[j].date.year>appointments[j+1].date.year)
            {
                temp=appointments[j];
                appointments[j]=appointments[j+1];
                appointments[j+1]=temp;
            }
            else if(appointments[j].date.month>appointments[j+1].date.month)
            {
                temp=appointments[j];
                appointments[j]=appointments[j+1];
                appointments[j+1]=temp;
            }
            else if(appointments[j].date.day>appointments[j+1].date.day)
            {
                temp=appointments[j];
                appointments[j]=appointments[j+1];
                appointments[j+1]=temp;
            }
            else if(appointments[j].date.time.hour>appointments[j+1].date.time.hour)
            {
                temp=appointments[j];
                appointments[j]=appointments[j+1];
                appointments[j+1]=temp;
            }
            else if(appointments[j].date.time.minute>appointments[j+1].date.time.minute)
            {
                temp=appointments[j];
                appointments[j]=appointments[j+1];
                appointments[j+1]=temp;
            }
        }
    }
}

void write_appointments(const char *file_name,Appointment_t appoinments[],int size)
{
    int i;
    FILE *outP;
    outP=fopen(file_name,"w");
    /* veriler dosyaya yazildi */
    for(i=0;i<size;i++)
    {
        fprintf(outP,"%2d %2d %2d ",appoinments[i].date.year,
                                    appoinments[i].date.month,
                                    appoinments[i].date.day);
        fprintf(outP,"%2d:%2d ",    appoinments[i].date.time.hour,
                                    appoinments[i].date.time.minute);
        fprintf(outP,"%d%d ",       appoinments[i].people.id_no.first_half,
                                    appoinments[i].people.id_no.second_half);
        fprintf(outP,"%s %s ",      appoinments[i].people.name,
                                    appoinments[i].people.surname);
        if(appoinments[i].people.gender)
            fprintf(outP,"F\n");
        else fprintf(outP,"M\n");
    }
```

```
    fclose(outP);
}

/* kisi ile kendinden sonra gelen herhangi biri arasinda zamanlar arasi */
/*  full esitlik var ise kendisi yedeklenecek yere alinmaz ve sonrakini alir*/
/* esik kisilerden 1ini yada 3esit varsa sadece sonuncuyu listemize aliriz */
int check_appointments(Appointment_t appointments[],int size)
{

    Appointment_t app2[100];
    int i,j,t=0;
    int equal=0;
    int find;

    for(i=0;i<size;i++)
    {
        find=0; /* esitlik yok */
        for(j=i;j<size;j++)
        {
           /* printf("i=%d - j=%d -",i,j); */

            equal=0;
            /* kendisini ingore ederiz  */
            if(appointments[i].date.year==appointments[j].date.year && i!=j)
                equal++;
            if(appointments[i].date.month==appointments[j].date.month && i!=j)
                equal++;
            if(appointments[i].date.day==appointments[j].date.day && i!=j)
                equal++;
            if(appointments[i].date.time.hour==appointments[j].date.time.hour && i!=j)
                equal++;
            if(appointments[i].date.time.minute==appointments[j].date.time.minute && i!=j)
                equal++;
            /* printf("%d",equal); */

            /* 5'te5 esitlik varsa bilgi dondur */
            if(equal==5)
                find=1; /* printf("Find = %d\n",find); */
        }

        /* esit degilse listeye yaz */
        if(find!=1)
        {
           app2[t]=appointments[i];
           t++; /* yeni kisi-randevu adedi */
        }
    }
    /* struct assignment */
    appointments=app2;

    return t;
}
/* HW08_HASAN_MEN_131044009_part2.c SONU */
```