```
/*############################################################################*/
/*HW08_part2.c                                                                */
/*_______________________________                                             */
/*Written by Mustafa Akilli on April 20, 2015                                 */
/*                                                                            */
/*Description                                                                 */
/*___________                                                                 */
/*  Appointments of a doctor                                                  */
/*Inputs:                                                                     */
/*  -People.txt                                                               */
/*  -AppointmentReqs.txt                                                      */
/*Outputs:                                                                    */
/*  -Appointments.txt                                                         */
/*############################################################################*/
/*                                                                            */
/*----------------------------------------------------------------------------*/
/*                            Includes                                        */
#include <stdio.h>
#include <string.h>
/*----------------------------------------------------------------------------*/
/*                             Define                                         */

/*----------------------------------------------------------------------------*/
/*                             Enumeration                                    */
typedef enum{M,F}Gender_t;
typedef enum{MONTHSZERO,JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST,
SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER}Months_t;
/*----------------------------------------------------------------------------*/
/*                             Structure                                      */
typedef struct
{
    int first_half;
    int second_half;
} TCId_no_t;

typedef struct
{
    TCId_no_t id_no;
    char name[30];
    char surname[30];
    Gender_t gender;
}People_t;

typedef struct
{
    int hour;
    int minute;
}Time_t;

typedef struct
{
    int year;
    Months_t month;
    int day;
    Time_t time;
}Date_t;

typedef struct
{
    People_t people;
    Date_t date;
}Appointment_t;
/*----------------------------------------------------------------------------*/
/*                             Functions                                      */
int get_people(const char *file_name, People_t people[], int max_size);
int get_appointments(const char *file_name, Appointment_t appointments[], int
max_size);
void write_names(Appointment_t appointments[], int size_app, const People_t
people[], int size_people);
int check_appointments(Appointment_t appointments[], int size);
void sort_appointments(Appointment_t appointments[], int size);
void write_appointments(const char *file_name, Appointment_t appointments[], int
```

```c
size);
/*-----------------------------------------------------------------------------*/

int
main(void)
{
    People_t people[5];
    Appointment_t appointments[5];
    int size_people,size_app,new_size_app;

    size_people=get_people("People.txt",people,5);
    size_app=get_appointments("AppointmentReqs.txt",appointments,5);

    write_names(appointments,size_app,people,size_people);

    new_size_app=check_appointments(appointments,size_app);

    sort_appointments(appointments,new_size_app);

    write_appointments("Appointments.txt",appointments,new_size_app);

    return 0;
}

/*  Reads all people from the file and record them into the array.
    Returns number records read.                                               */
int get_people(const char *file_name, People_t people[], int max_size)
{
    char charter;
    int status,i,people_size;

    FILE* inp;

    inp=fopen(file_name,"r");

    status=fscanf(inp,"%6d",&people[0].id_no.first_half);

    for(i=0;status!=EOF;++i)
    {
        fscanf(inp,"%6d",&people[i].id_no.second_half);
        fscanf(inp,"%s",people[i].name);
        fscanf(inp,"%s",people[i].surname);
        fscanf(inp,"%s",&charter);
        people[i].gender=charter;

        status=fscanf(inp,"%6d",&people[i+1].id_no.first_half);
    }

    fclose(inp);

    people_size=i;

    return people_size;


}

/*  Reads all appointment records from the file and
    record them into the array leaving name and
    surname fields unassigned.
    Returns number records read.                                               */
int get_appointments(const char *file_name, Appointment_t appointments[], int
max_size)
{
    int status,i,temp,month,appointments_size;
    char temp_char;

    FILE* inp;

    inp=fopen(file_name,"r");

    status=fscanf(inp,"%6d",&appointments[0].people.id_no.first_half);
    for(i=0;status!=EOF;++i)
    {
        fscanf(inp,"%6d",&appointments[i].people.id_no.second_half);
        fscanf(inp,"%d",&appointments[i].date.year);
        fscanf(inp,"%d",&month);
        appointments[i].date.month=month;
        fscanf(inp,"%d",&appointments[i].date.day);
        fscanf(inp,"%d",&appointments[i].date.time.hour);
        fscanf(inp,"%c",&temp_char);
        fscanf(inp,"%d",&appointments[i].date.time.minute);

        status=fscanf(inp,"%6d",&appointments[i+1].people.id_no.first_half);
    }

    fclose(inp);

    appointments_size=i;

    return appointments_size;
}

/*  Fills the name and surname
    fields of people fields of appointments.                                    */
void write_names(Appointment_t appointments[], int size_app, const People_t
people[], int size_people)
{
    int i;

    for(i=0;i<size_app;++i)
    {
        strcpy(&(appointments[i].people.name[0]),&people[i].name[0]);
        strcpy(appointments[i].people.surname,people[i].surname);
        appointments[i].people.gender=people[i].gender;
    }

}

/*  Considers all appointment requests,
    deletes the rejected ones and
    returns the new size as the return value.                                   */
int check_appointments(Appointment_t appointments[], int size)
{
    int j,i,temp_j;

    for(i=0;i<size;++i)
    {
        for(j=i+1;j<size;++j)
        {

            if(appointments[i].date.year==appointments[j].date.year &&
                appointments[i].date.month==appointments[j].date.month &&
                appointments[i].date.day==appointments[j].date.day &&
                appointments[i].date.time.hour==appointments[j].date.time.hour &&
                appointments[i].date.time.minute==appointments[j].date.time.minute)
            {
                temp_j=j;
                while(temp_j<size-1)
                {
                    appointments[temp_j]=appointments[temp_j+1];
                    ++temp_j;
                }
                --j;
                --size;
            }
        }
    }

    return size;
}
```

```c
/*  Sort the array with respect to the date of the appointment.                 */
void sort_appointments(Appointment_t appointments[], int size)
{
    int i,j;
    Appointment_t temp_appointments[1];

    for(j=1; j<=size-1; ++j)
        for(i=0; i<=size-2; i++)
            if(appointments[i].date.year>appointments[i+1].date.year)
            {
                temp_appointments[0]=appointments[i];
                appointments[i]=appointments[i+1];
                appointments[i+1]=temp_appointments[0];
            }

            else if(appointments[i].date.year<appointments[i+1].date.year)
            {

            }

            else
                {
                    if(appointments[i].date.month>appointments[i+1].date.month)
                    {
                        temp_appointments[0]=appointments[i];
                        appointments[i]=appointments[i+1];
                        appointments[i+1]=temp_appointments[0];
                    }

                    else if(appointments[i].date.month<
                            appointments[i+1].date.month)
                    {

                    }

                    else
                    {
                        if(appointments[i].date.day>
                           appointments[i+1].date.day)
                        {
                            temp_appointments[0]=appointments[i];
                            appointments[i]=appointments[i+1];
                            appointments[i+1]=temp_appointments[0];
                        }

                        else if(appointments[i].date.day<
                                appointments[i+1].date.day)
                        {

                        }

                        else
                        {
                            if(appointments[i].date.time.hour>
                               appointments[i+1].date.time.hour)
                            {
                                temp_appointments[0]=appointments[i];
                                appointments[i]=appointments[i+1];
                                appointments[i+1]=temp_appointments[0];
                            }

                            else if(appointments[i].date.time.hour<
                                    appointments[i+1].date.time.hour)
                            {

                            }

                            else
                            {
                                if(appointments[i].date.time.minute>
                                   appointments[i+1].date.time.minute)
                                {
                                    temp_appointments[0]=appointments[i];
                                    appointments[i]=appointments[i+1];
                                    appointments[i+1]=temp_appointments[0];
                                }
                            }
                        }
                    }
                }
}

/*  Writes all appointments to a text file.                                    */
void write_appointments(const char *file_name, Appointment_t appointments[], int
size)
{
    int i;

    FILE* outp;

    outp=fopen(file_name,"w");

    for(i=0;i<size;++i)
    {
        fprintf(outp,"%d ",appointments[i].date.year);
        fprintf(outp,"%d ",appointments[i].date.month);
        fprintf(outp,"%d ",appointments[i].date.day);
        fprintf(outp,"%d:",appointments[i].date.time.hour);
        fprintf(outp,"%d ",appointments[i].date.time.minute);
        fprintf(outp,"%d",appointments[i].people.id_no.first_half);
        fprintf(outp,"%d ",appointments[i].people.id_no.second_half);
        fprintf(outp,"%s ",appointments[i].people.name);
        fprintf(outp,"%s ",appointments[i].people.surname);
        fprintf(outp,"%c\n",appointments[i].people.gender);


    }

    fclose(outp);
}
/*##############################################################################*/
/*                              End of HW08_part2.c                             */
/*##############################################################################*/
```