```c
/*#############################################################################*/
/*Created by Emre Bayram 141044019 part2                                       */

#include<stdio.h>
#include<string.h>
#define MAX_SIZE 30

typedef enum {MALE,FEMALE}gender_t;
typedef enum {EMPTY,JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST,SEPTEMBER,
              OCTOBER,NOVEMBER,DECEMBER} months_t;

typedef struct{ int hour;
                int minute;
                }Time_t;
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 year;
                months_t month;
                int day;
                Time_t time;
                }Date_t;
typedef struct{ People_t people;
                Date_t date;
                }Appointment_t;
/*#############################################################################*/
/*                           For Abstact data types                            */
Time_t get_time(int hour,int minute)
{
    Time_t time;
    time.hour = hour;
    time.minute = minute;

    return time;
}

TCId_no_t get_tc(int firsthalf,int secondhalf)
{
    TCId_no_t tc;

    tc.first_half=firsthalf;
    tc.second_half=secondhalf;

    return tc;
}

Date_t get_date(int year, months_t months,int day, Time_t time)
{
    Date_t date;
    date.year=year;
    date.month=months;
    date.day=day;
    date.time=time;

    return date;
}
/*#############################################################################*/

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 size);
```

```c
void go_bacward(Appointment_t appointment[],int *size);

int main ()
{
    People_t people[MAX_SIZE];
    Appointment_t appointment[MAX_SIZE];
    char fpeople[]="people.txt";
    char fpeoplereqs[]="peoplereqs.txt";
    char foutput[]="output.txt";
    int size;
    int app_size;
    int i;
    size = get_people(fpeople,people,MAX_SIZE);
    app_size = get_appointments(fpeoplereqs,appointment,MAX_SIZE);

    if(app_size == size)
    {
        write_names(appointment,app_size,people,size);
        size = check_appointments(appointment,size);
        sort_appointments(appointment,size);
        write_appointments(foutput,appointment,size);

    }
    else printf("appointments size and people size are not the same please check it \n");

    return 0;
}

int get_people(const char *file_name, People_t people[], int max_size)
{
    int tc1,tc2;
    int i=0;
    int flag=1;
    int status;
    char name[30],surname[30];
    gender_t gender;
    char gndr;
    TCId_no_t id;

    FILE* inp;

    inp=fopen(file_name,"r");

    do{
        status=fscanf(inp,"%6d%5d %s %s %c\n",&tc1,&tc2,name,surname,&gndr);

        switch (gndr){
            case 'M' : gender = MALE;
                break;
            case 'F' : gender = FEMALE;
                break;

        }

        if(status == EOF || i == max_size)
            flag=0;
        else
        {
            id=get_tc(tc1,tc2);
            people[i].id_no=id;
            strcpy(people[i].name,name);
            strcpy(people[i].surname,surname);
            people[i].gender=gender;
            ++i;
        }
    }while(flag);

    fclose(inp);

    return  i;
}
```

```
int get_appointments(const char *file_name, Appointment_t appointments[],
                     int max_size)
{

    int year,month,day,hour,min,tc1,tc2;
    TCId_no_t tc;
    Time_t time;
    Date_t date;
    char dead;
    int i=0;
    int status;
    int flag=1;
    FILE* inp;

    inp=fopen(file_name,"r");

    do{
        status=fscanf(inp,"%6d%5d %d %d %d %2d%c%2d ",&tc1,&tc2,&year,&month,&day,&hour,&dead,&min);
        if(status == EOF || i == max_size)
            flag=0;
        else
        {
            time=get_time(hour,min);
            date=get_date(year,month,day,time);
            tc=get_tc(tc1,tc2);
            appointments[i].date=date;
            appointments[i].people.id_no=tc;
            ++i;
        }
    }while(flag);

    fclose(inp);

    return i;
}
void write_names(Appointment_t appointments[], int size_app,
                 const People_t people[], int size_people)
{
    int i;
    for(i=0;i<size_people && i<size_app;++i){
        strcpy(appointments[i].people.name,people[i].name);
        strcpy(appointments[i].people.surname,people[i].surname);
        appointments[i].people.gender=people[i].gender;
    }
}
int check_appointments(Appointment_t appointments[], int size)
{
    int i,j;
    for(i=0;i < size;++i){
        if(appointments[i].date.time.minute != 30 && appointments[i].date.time.minute != 0)
            {
                go_bacward(&appointments[i],&size);
            }
        }
    for(i=0;i<size;++i)
        for(j=i+1;j<size;++j)
        {
            if(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 &&
               appointments[i].date.year == appointments[j].date.year &&
               appointments[i].date.month == appointments[j].date.month)
            {
                go_bacward(&appointments[i],&size);
            }
        }

    return size;
}

void sort_appointments(Appointment_t appointments[], int size)
{
```

```
    int i,j;
    Appointment_t temp;
    for(j=0;j<size;++j)
        for(i=1;i<size;++i){
            if(appointments[i-1].date.year > appointments[i].date.year)
            {
                temp = appointments[i-1];
                appointments[i-1] = appointments[i];
                appointments[i] = temp;
            }
            else if (appointments[i-1].date.month > appointments[i].date.month)
            {
                temp = appointments[i-1];
                appointments[i-1] = appointments[i];
                appointments[i] = temp;
            }
            else if (appointments[i-1].date.day > appointments[i].date.day)
            {
                temp = appointments[i-1];
                appointments[i-1] = appointments[i];
                appointments[i] = temp;
            }
            else if (appointments[i-1].date.time.hour > appointments[i].date.time.hour)
            {
                temp = appointments[i-1];
                appointments[i-1] = appointments[i];
                appointments[i] = temp;
            }
            else if (appointments[i-1].date.time.minute > appointments[i].date.time.minute)
            {
                temp = appointments[i-1];
                appointments[i-1] = appointments[i];
                appointments[i] = temp;
            }
        }
}

void write_appointments(const char *file_name, Appointment_t appointments[],
                        int size)
{
    char gndr;
    int i;
    FILE *outp;

    outp = fopen(file_name,"w");

    for(i=0;i<size;++i){
        switch (appointments[i].people.gender){
            case MALE : gndr = 'M';
                break;
            case FEMALE : gndr = 'F';
                break;
        }
        fprintf(outp,"%d%d %-10s %-10s %c   %d/%u/%d    %2d:%2d\n",appointments
[i].people.id_no.first_half,
                                                            appointments[i].people.id_no.second_half,
                                                            appointments[i].people.name,
                                                            appointments[i].people.surname,
                                                            gndr,
                                                            appointments[i].date.year,
                                                            appointments[i].date.month,
                                                            appointments[i].date.day,
                                                            appointments[i].date.time.hour,
                                                            appointments[i].date.time.minute);
    }
}

void go_bacward(Appointment_t appointment[],int *size)
{
    int i;
    for(i=0;i<(*size);++i)
    {
```

```
        appointment[i]=appointment[i+1];
    }
    --(*size);
}
```