```c
#include "part1.h"

const char *Records_bin="Records.bin";
const char *Patients_xml="Patients.xml";
const char *Delete_txt="Delete.txt";
const char *Records_xml="Records.xml";
const char *Appointments_csv="Appointments.csv";
const char *Parameters_txt="Parameters.txt";

/* Reads all appointments in the records file into a dynamically
    allocated fully-filled array and returns the array. */
Appointment_t* getRequests(const Files_t* files, int* size)
{
    Appointment_t *Appointments;

    FILE *binary_input_file;

    binary_input_file=fopen(files->records_file_n,"rb");

    if(binary_input_file == NULL)
    {
        printf("ERROR!! Binary file could not be opened to read.\n");
        exit(1);
    }

    fread(size,sizeof(int),1,binary_input_file);

    Appointments=(Appointment_t*)malloc((*size)*sizeof(Appointment_t));

    fread(Appointments,sizeof(Appointment_t),*size,binary_input_file);

    fclose(binary_input_file);

    return Appointments;
}

/* Writes all appointments in the input array
    to readable records file in the described format */
void write_appointments(Appointment_t appointments[], int size, const Files_t* files)
{
    FILE *xml_output_file;

    int i;

    xml_output_file=fopen(files->readable_records_file_n,"w");

    if(xml_output_file == NULL)
    {
        printf("ERROR!! .xml file could not be opened to write.\n");
        exit(1);
    }

    fprintf(xml_output_file,"<Size>%d</Size>\n",size);
    fprintf(xml_output_file,"<Records>\n");

    for(i=0;i<size;++i)
    {
        fprintf(xml_output_file,"   <Appointment>\n");
        fprintf(xml_output_file,"       ");
        fprintf(xml_output_file,"<app_id>%d</app_id>\n",appointments[i].app_id);
        fprintf(xml_output_file,"       ");
        fprintf(xml_output_file,"<patient_id>%d</patient_id>\n",appointments[i].patient_id);
        fprintf(xml_output_file,"       ");
        fprintf(xml_output_file,"<hour>%d</hour>\n",appointments[i].hour);
        fprintf(xml_output_file,"   </Appointment>\n");
    }

    fprintf(xml_output_file,"</Records>\n");

    fclose(xml_output_file);
}
```

```
/* Takes the arguments of main() as input parameter and
    returns used input file names and working hours as output parameters*/
void get_main_arguments(int argc, char *argv[], Working_hours_t* hours, Files_t*
files)
{
    int check_file_Records=DEFAULT_FILE;
    int check_file_Patients=DEFAULT_FILE;
    int check_file_Delete=DEFAULT_FILE;
    int check_file_Readable_Records=DEFAULT_FILE;
    int check_file_Accepted_Appointments=DEFAULT_FILE;
    int check_file_Parameters=DEFAULT_FILE;
    int check_start_work=DEFAULT_FILE;
    int check_end_work=DEFAULT_FILE;

    char *token[ARRAY_SIZE];
    const char s[2] = "-";
    int i;

    /* If the file names are changed which found that the change */
    for(i=1;i<argc;i+=2)
    {
        token[i] = strtok(argv[i],s);
    }

    /* If the file names are changed, use the changed file name*/
    for(i=1;i<argc;i+=2)
    {
        if(*token[i]=='r')
        {
            files->records_file_n=argv[i+1];
            check_file_Records=CHANGE_FILE;
        }

        if(*token[i]=='p')
        {
            files->patients_file_n=argv[i+1];
            check_file_Patients=CHANGE_FILE;
        }

        if(*token[i]=='d')
        {
            files->delete_file_n=argv[i+1];
            check_file_Delete=CHANGE_FILE;
        }

        if(*token[i]=='x')
        {
            files->readable_records_file_n=argv[i+1];
            check_file_Readable_Records=CHANGE_FILE;
        }

        if(*token[i]=='c')
        {
            files->accepted_appo_file_n=argv[i+1];
            check_file_Accepted_Appointments=CHANGE_FILE;
        }

        if(*token[i]=='t')
        {
            files->parameters_file_n=argv[i+1];
            check_file_Parameters=CHANGE_FILE;
        }

        if(*token[i]=='s')
        {
            hours->start=atoi(argv[i+1]);
            check_start_work=CHANGE_FILE;
        }

        if(*token[i]=='e')
        {
            hours->end=atoi(argv[i+1]);
```

```
            check_end_work=CHANGE_FILE;
        }
    }

    /* If the file names are not changed, use the default file name*/
    if(check_file_Records==DEFAULT_FILE)
    {
        files->records_file_n=Records_bin;
    }

    if(check_file_Patients==DEFAULT_FILE)
    {
        files->patients_file_n=Patients_xml;
    }

    if(check_file_Delete==DEFAULT_FILE)
    {
        files->delete_file_n=Delete_txt;
    }

    if(check_file_Readable_Records==DEFAULT_FILE)
    {
        files->readable_records_file_n=Records_xml;
    }

    if(check_file_Accepted_Appointments==DEFAULT_FILE)
    {
        files->accepted_appo_file_n=Appointments_csv;
    }

    if(check_file_Parameters==DEFAULT_FILE)
    {
        files->parameters_file_n=Parameters_txt;
    }

    if(check_start_work==DEFAULT_FILE)
    {
        hours->start=START_WORK;
    }

    if(check_end_work==DEFAULT_FILE)
    {
        hours->end=END_WORK;
    }
}

/* Writes file names and working hours
    to Parameters file in the described format.*/
void print_parameters(const Files_t* files, const Working_hours_t* hours)
{
    if(!DEBUG)
    {
        printf("Records File Name = %s\n",files->records_file_n);
        printf("Patients File Name = %s\n",files->patients_file_n);
        printf("Delete File Name = %s\n",files->delete_file_n);
        printf("Readable Records File Name = %s\n",files->readable_records_file_n);
        printf("Accepted Appointments File Name = %s\n",files->accepted_appo_file_n);
        printf("Parameters File Name = %s\n",files->parameters_file_n);
        printf("Start Work Hour = %d\n",hours->start);
        printf("End Work Hour = %d\n",hours->end);
    }
}
/*############################################################################*/
/*                            End of HW10_part1.c                             */
/*############################################################################*/
```