Milestone – 3 (Code: weight 2.5%)

Milestone-3 completes the veterinary clinic system with the addition of appointment management functionality and the importing of patient and appointment information from data text files. You will need to create additional data structures along with several functions that will be responsible for carrying out the new appointment management menu options that perform specific tasks in the management of appointment data.

Specifications

Like Milestone-2, it is important to note that this code will not compile until you copy your work from Milestone-2 into the provided files for Milestone-3. You will also need to create some new data types, as well as define the mandatory uncoded function prototypes and definitions that are new to Milestone-3.

Three ***clinic*** module/library functions have been completely supplied for you: "***displayScheduleHeader***", "***displayScheduleData***", and "***menuAppointment***", along with two function prototypes: "***importPatients***" and "***importAppointments***" (responsible for data import). These functions provide you with the necessary framework to get started in this final milestone.

In this milestone, **it is expected you will create additional functions** as you see fit to help you get the job done. It is also expected you will **follow and adhere to the structured design principles** (Functions | Introduction to C (sdds.ca)).

When you create your own functions, be sure to organize them into the established commented sections for each module/library so you can easily find and maintain them:

Core
- User Interface
- User Input
- Utility

Clinic
- Display Functions
- Menu & Item Selection
- Utility
- User Input
- File

Recommended Approach

You need to get the project ready for development so it can compile and test your new functions with actual data. It is recommended you develop the new components in the following sequence:

1. Create/define the new data types: ***Time***, ***Date***, and ***Appointment***

2. Code the ***importPatients*** and ***importAppointments*** function definitions. The ***main*** function requires these functions to be working so the application can start with data preloaded.

3. Create the function prototypes and empty function definitions (stubs) for those functions called within the ***menuAppointment*** function.

Clinic Module

In this milestone, you will be working almost exclusively with the ***clinic*** module unless you find reason to add more generalized functions to the ***core*** module.

Data Structures

A few new data structures will be needed to represent the appointment information. This includes a ***Time***, a ***Date***, and an ***Appointment*** type. These types need to be defined in the ***clinic.h*** file (**review the comments** in the ***clinic.h*** file for placement).

Hint

Review the following resources to help you determine the members for these new data types:

- Function: "***displayScheduleData***" (clinic.c file)
- Data file: "***appointmentData.txt***" (see description in next section)

Data Files

There are two data files you will need to import into the application. One contains patient data, while the other contains appointment information. These files are provided along with the project files on GitHub.

patientData.txt

The data fields for the patient data are **separated by a pipe (|)** character in the following order:

- Patient number
- Patient name
- Contact phone description
- Contact phone number

appointmentData.txt

The data fields for the appointment data are **separated by a comma (,)** character in the following order:

- Patient number
- Appointment year
- Appointment month
- Appointment day
- Appointment hour
- Appointment minute

Functions

Data Import (text files)

You must create the "***importPatients***" and "***importAppointments***" function definitions (in the ***clinic.c*** file) that will read-in the text data and store the data to their respective ***struct Patient*** and ***struct Appointment*** arrays. Review the ***main*** function to see how they are called. The function prototypes for these functions have been provided for you located in the ***clinic.h*** header file.

Here is a brief overview of how these functions should work. Both import functions should do the following:

- Read the data file (file name is provided in the first argument)
- Store the data to the second argument, an array of the respective struct data type
- Respect the array size specified by the 3rd argument even when the data file has more records
- Must return the total number of records read from the file and stored to the array, which may be less than the total number of records in the file

Appointment Management

To get you started, the appointment management menu and display functions have been provided for you:

- ***menuAppointment***
- ***displayScheduleHeader***
- ***displayScheduleData***

Hints

- Create temporary empty function shells for the following functions:
    - ***viewAllAppointments***
    - ***viewAppointmentSchedule***
    - ***addAppointment***
    - ***removeAppointment***

  All these functions are called from the "***menuAppointment***" function and will need to exist for you to be able to compile your code.

- Review the sample output text file (***a1ms3_output.txt***) that came with the project files to see how these menu options should work

viewAllAppointments

- Carefully review the sample output text file (***a1ms3_output.txt***) that came with the project files and notice the order of the data is not presented in the order it is positioned in the appointments array.
- At some point in the logic for this menu option, you will need to call the "***displayScheduleHeader***" and "***displayScheduleData***" functions

viewAppointmentSchedule

- This process should display only the appointments scheduled for a specific date. Therefore, the user must be prompted for a specific date (year, month, and day) prior to displaying the results.
- Date input prompting and validations must accurately determine the number of days in a given month and accommodate leap years
- Carefully review the sample output text file (***a1ms3_output.txt***) that came with the project files and notice the order of the data is not presented in the order it is positioned in the appointments array.
- At some point in the logic for this menu option, you will need to call the "***displayScheduleHeader***" and "***displayScheduleData***" functions

addAppointment

- This process should test if there is an available element in the appointments array for a new appointment to be added. Available appointments can be determined by testing the patient number which must be less than 1 to indicate an empty/available element.
- Validation of the entered patient number must be performed
- Appointment times must adhere to the following rules:
    - **Operation hours** are determined based on macro's that define the start and end hours so they can be modified in one place without affecting the code logic
    - The effective appointment start times must fall within the inclusive hour range defined by the macros mentioned above (example: 9:00 – 16:00, so the last valid appointment time can be 16:00)
    - The entered minute value must align with the set appointment minute interval which should be represented as a macro so it can be modified in a single place and not affect the code logic (example: if appointments are in 15-minute intervals it is only possible to have appointments starting at 0, 15, 30, or 45 minutes on the hour)
    - Only ONE appointment can occupy a time slot for the given year, month, and day (**no double booking or overlap is allowed**)
- Again, carefully review the sample output text file (***a1ms3_output.txt***) that came with the project files to see how this process should work.

removeAppointment

- Appointments are removed by patient number and a specific date (year, month, and day).
- The entered patient number must be validated as a patient who has not been removed from the ***patients*** array or the removal should be denied (this will ensure that past appointment data will remain intact).
- When an appointment match occurs (based on the entered patient number and date), the patient information details should be displayed, and user confirmation must be received to remove the appointment
- To mark an appointment as removed, the appropriate appointments array element must be set to a safe empty state.
- Again, carefully review the sample output text file (***a1ms3_output.txt***) that came with the project files to see how this process should work.

A1-MS3: Sample Output

Please review the provided text file for this milestone (on GitHub) "***a1ms3_output.txt***"

Reflection (Weight: 2.5%)

Academic Integrity

It is a violation of academic policy to copy content from the course notes or any other published source (including websites, work from another student, or sharing your work with others).