Assignment 1 - Reflection

Full Name  : Joseph Mwamba-Mukuna

Student ID#: 163887216

Email      : jmwamba-mukuna@myseneca.ca

Section    : NRA

Authenticity Declaration:

I declare this submission is the result of my own work and has not been

shared with any other student or 3rd party content provider. This submitted

piece of work is entirely of my own creation.

1. Most of the functions that were created were done in order to, having an easier with the implementation and modification of the appointment struct arrays.  
   *void sort(struct Appointment app[], int max)* is the first function I made in order to display the appointments in ascending order. Arguably one of the more important functions since, for the functions that required viewing I was important that they appeared ordered and proper. It’s incredibly valuable as having to write a sorting algorithm for every different “view” type function wouldn’t proven time consuming. The idea of converting the different appointment times into one massive int before sorting came from the application Discord and remembering how UNIX time works. I believed it would be easiest to convert the times into seconds just like in UNIX then compare them and arranged them using the selection sort method.

int timeSlotCheck(struct Date date, struct Time time, struct Appointment\* app, int maxAppointments) is the next function I thought to be useful to the demands of this milestone, since with modifications done with the appointments such as adding an appointment. It was necessary to have a module to check through the array and determine if there was a similar appointment with the same time frame.

int openSlot(struct Appointment\* app, int maxAppointments) this function was created in order to find a empty index in the array to add an appointment, by parsing through the array in question and returning a value of 1 or 0, if a spot was open or not. Once the open slot was found, it spot was saved then filled the new appointments. This function was extremely useful for the addAppointment function since naturally to add an appointment you must first check if there is an open spot for the patient then confirm it by saving it there.

1. The reason there was so little use of the scanf function in this milestone, is not only due to the previous modules and milestones doing the heavy lifting then implementing them in newer functions via function calls. But also due to the variety of other methods used to better acquire user input through functions like getchar, fgets, etc. I feel this was done purposely to limit some of the issues and limitations that can be brought up with scanf, but also to show that are better and more efficient wys to get user input and have them work accordingly. A good example of this is in the inputCString function, instead of using scanf along with multiple condition statements and format specifiers. I found it much simpler to use getchar so it can read the characters and so that it would adhere to the constraints that I needed at times.
2. *void sort(struct Appointment app[], int max)* I had placed this function in both the viewAllAppointments function as well as the viewAppointmentSchedule function. Due to the fact that when viewing the appointments or patients in the console, they needed to be properly ordered to not only match the output but for it to appears neater and more organized for the user and or programmer who chooses to visit this code. int timeSlotCheck(struct Date date, struct Time time, struct Appointment\* app, int maxAppointments) this function was used in the addAppointment function in order to have an easy time checking if the date and time selected by the user was available to schedule or not. int openSlot(struct Appointment\* app, int maxAppointments) this function was once again in the addAppointment function as a key element, as it was necessary to be able to determine and find if a slot was available to fit an appoint