MP4: Struct

Objectives:

- Write functions that use file I/O and structs.
- (Optional) Write a function to sort an array of structs.

Tasks:

1. Open a text file and create an array of structs.
2. Print a struct.
3. Print an array of structs, in order.
4. Print an array of structs, sorted by one field.

For problem sessions, you may work with other students, but every student will be graded individually, based on his/her own submitted work. Every student must submit the solution.

These instructions will assume that you are using the CLion integrated development environment. If you are using another platform, the steps will be slightly different, but you should be able to do everything described in this assignment. The C source code file should compile and run in any environment.

Task 0: Setup

In CLion, create a New Project. Name it something appropriate, like **mp4**. Download all files from the ZyLab: **15.4 MP4: Structs**. Unzip the file and copy all files into your new project: **main.c, Appointment.h, Appointment.c, monday.txt, friday.txt**.

Finally, set the Working Directory for the program.[1] Click on the pull-down menu to the right of the hammer icon in the top right corner, and select **Edit Configurations...** In the box next to **Working Directory**, click on the folder and select the project folder on your machine.

Do not modify any file except **Appointment.c**.

The program will compile and run as is. The main function will prompt for a file name and for a task number. The task number corresponds to one of the tasks below.

[1] If you are working on a Linux workstation, this step is not necessary. Just put the text files in the same directory where you are compiling your executable.

Task 1: Read a file and create a struct array

The **readFile** function will open a text file and read information, storing into an array of structs. The file contains appointment information for a veterinary clinic. The file contains one line per appointment with information in this order:

- time, in hh:mm 24-hr format
- owner's name, one word (no spaces), up to 24 characters
- pet's name, one word (no spaces), up to 24 characters
- pet type, one word (no spaces), up to 24 characters

There will be no more than 20 appointments in a file. There are two struct types defined in Appointment.h: **struct appt** (also known as **Appointment**) and **struct time**.

For this task: (1) Open the file. (2) Allocate an array of Appointment values to hold the data. (3) Read the information into the array, in the same order as the file. (4) Close the file. (5) Set the parameter indicating how many appointments were read. (6) Return the array.

If the file cannot be opened, return NULL.

Upload to ZyLab for testing and grading. (Task 1 tests will call **readFile** and check the returned values.)

Task 2: Print a struct

The **printAppointment** function takes a *pointer* to a *const struct appt* and prints the information from the struct value to the standard output stream (stdout). The format must look exactly like the following:


```
hh:mm Owner / Pet (type)
```

There must be a linefeed printed at the end. There is exactly one space between each part.

The time must print two digits for hours and mins. If the number is only one digit, a leading zero is printed. The format string to do this is "%02d" -- this says: print a decimal integer using at least two characters, and fill unused positions to the left with a zero. In other words, the value 3 will be printed as `03`. This is just to make the output look nice, since all of the times will take up the same number of character in the functions below.

For Task 2, the **main** function will call **printAppointment** to print the first appointment in the array, and the ZyBook test will check this printed output.

Task 3: Print an array of structs

The **printAll** function prints an array of Appointment values. The function takes an array (pointer) and an integer that indicates the size of the array. Each array element is printed using **printAppointment**.

Task 4: Print an array of structs in sorted order

The **printAllByPet** function prints an array of Appointment values. The function takes an array (pointer) and an integer that indicates the size of the array. This time, the appointments must be printed in order of pet name, alphabetically. (Case doesn't matter: use **strcasecmp**.)

The array pointer passed in is declared as const, so you can't directly sort it. You may create a copy of the array and use one of the sorting algorithms discussed in class. (I recommend selection sort.) Or you can search through the array for the next minimum element, and print the value when you find it. The implementation is up to you. (You may not use the **qsort** function from the C standard library.)

The Task 4 tests only look at the printed output. You are not required to create a sorted array.

GRADING

Task 1: 60 pts
Task 2: 10 pts
Task 3: 20 pts
Task 4: 10 pts