CS1050 – Lab 9

Spring 2020

Concepts to Practice

- File Processing

Submission Information

Submit this assignment by following the instructions given by your TA. SUBMIT ONLY the .c file (no a.out or executable file is required). All of the lab assignments must be submitted before the end of the lab using the lab code given by the TA.

Use the following submit command:

```
mucs submit <class> <assignment_type> <filename>
```

For example:

```
mucs submit 1050 lab x-lab9.c
```

Filenames must be: *sectionletter-lab9.c* (include your respective lab section, e.g., a-lab9.c).

Description

So, we are all going stir-crazy from being stuck in our homes, and so my mind turns to Dungeons and Dragons. As it turns out, my character (Ahote) and my brother's character (Gary) are stuck together due to the Corona Virus as well. Since they aren't real, the best solution is for them to attack each other (don't try this at home!).

For the lab assignment, you will read data from two data files. The format of each file is the same and will look something like this:

```
Ahote 18 75 11 8
```

The first part is the name of the character (Ahote, in this case), the next item is that character's armor class (18), his hit points (75), his hit bonus (11) and his damage bonus (8). For your lab, you will read data from each of the following two files:

```
/group/cs1050/data/ahote.dat
/group/cs1050/data/gary.dat
```

If it is helpful, you can copy these files locally to your home directory on tc.rnet.missouri.edu with commands like this:

```
cp /group/cs1050/data/ahote.dat .
cp /group/cs1050/data/gary.dat .
```

This will copy each data file to the current directory. You are welcome to look at the data in these files using "cat" or "vim" or whatever you like.

Once you have read all of the data, you should print it out in readable format. Then, you will call a function (written by me) that has a prototype that looks like this:

```
void Fight(     char * combatant1, int ac1, int hp1, int hitbonus1, int damagebonus1,
                char * combatant2, int ac2, int hp2, int hitbonus2, int damagebonus2);
```

You will need to include my header file called "fight.h" in the same way as you usually include Standard C Library headers like "stdio.h". When you compile your program (e.g, "compile x-lab9.c"), you will automatically get my code linked in with your program.

Bonus

Yep, this program includes the possibility of getting bonus points again! Hooray! All you have to do is remember my lecture about command-line processing (or look it up if you have time in lab). To get the bonus points, you must not “hard-code” the name of the two data files. Instead, you must take each data file as a command-line argument. If you do the bonus, you must also check that the user has put in the correct number of arguments, and that the specified files can actually be read.

Sample Output (shown using optional Bonus command-line parameters)

```
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>cp /group/cs1050/data/ahote.dat .
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>cp /group/cs1050/data/gary.dat .
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>compile x-lab9.c
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>./a.out
*** Syntax:
./a.out combatant1_file combatant2_file

jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>./a.out asdf asdf
** Error:  Could not open asdf
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>./a.out ahote.dat asdf
** Error:  Could not open asdf
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>./a.out asdf gary.dat
** Error:  Could not open asdf
jer676@clark-r630-login-node907:/home/jer676/CS1050/SP2020/labs/lab9>./a.out ahote.dat gary.dat
Name=Ahote,AC=18,HP=75,ToHitBonus=11,DamageBonus=8
Name=Gary,AC=18,HP=61,ToHitBonus=1,DamageBonus=1
**** Combat between Ahote and Gary ****
                Ahote misses Gary
                Gary misses Ahote
        Ahote hits Gary for 14 damage
                Ahote=75, Gary=47
                Gary misses Ahote
        Ahote hits Gary for 15 damage
                Ahote=75, Gary=32
                Gary misses Ahote
        Ahote hits Gary for 10 damage
                Ahote=75, Gary=22
                Gary misses Ahote
        Ahote hits Gary for 9 damage
                Ahote=75, Gary=13
        Gary hits Ahote for 5 damage
                Ahote=70, Gary=13
                Ahote misses Gary
                Gary misses Ahote
                Ahote misses Gary
                Gary misses Ahote
        Ahote hits Gary for 10 damage
                Ahote=70, Gary=3
                Gary misses Ahote
        Ahote hits Gary for 18 damage
                Ahote=70, Gary=-15
Ahote wins!
**** Combat ends ****
```

Guidelines for Grading Lab 9
40 Points Possible (+5 Bonus)

General

If your program does not compile or produce any input/output (I/O) because most of the source code is commented out then your lab will receive a grade of ZERO POINTS. Further, if your program does not actually follow the specifications, but merely prints out lines that make it appear to follow the specifications, you will receive a grade of ZERO POINTS. For partial credit your C program must not only compile but also produce some valid I/O that meets the lab specifications.

You program is expected to have a comment header at the top that includes your name, pawprint, the course you are taking, and the lab that you are solved (e.g., "Lab 9"). Your code should be nicely indented. **You will lose up to 10 points if you do not meet these basic requirements**.

10 points: Your code opens each file correctly.
5 points: Your code correctly closes each before the program ends.
10 points: Your code reads and prints each data field correctly.
10 points: Your program is able to use the provided Fight() function, calling it with correct parameters.
5 points: Your output matches the sample output.
BONUS 5 points: Your code accepts command-line parameters specifying which files to open (and in which order). It checks whether the user has input enough information on the command-line and whether the provided file names can be used to get data.