Homework 5, TCSS 333 Autumn 2015
Due: Monday, December 14, 9:00 a.m.

OBJECTIVE

The objective of this assignment is to give you more practice with using structures and binary random access files.

ASSIGNMENT SUBMISSION

To get credit for this assignment, you must
- ✓ write a multi-file program in C
- ✓ submit your project as a tar file through Canvas exactly as instructed (naming, compatibility, etc.)
- ✓ submit your assignment on time

PROBLEM STATEMENT

For this assignment, you need to write a program that maintains a database for a local animal rescue as a binary file, `animals.dat` (provided with the assignment). The program needs to support operations such as (1) adding an animal, (2) deleting an animal, (3) updating animal's information, (4) searching for an animal, and finally (5) writing all animals' information to an ASCII text file called `animals.csv`. These operations need to be presented to the user as options (use the numbers given above) and the program can only terminate when the user selects to do so (0). All the operations need to be performed directly on the database file, without storing all animals into a data structure, such as an array. The program can only store one animal at a time and a few primitives required by the processing logic.

An animal should be stored and processed using the following structure:

```
struct animal {
    short int id;
    char name[20];
    char species[35];
    short int age;
    double weight;
};
```

Output Files
Your program should produce two output files: an updated `animals.dat` file, according to the user's operations, and an ASCII output file `animals.csv` that lists one animal per line and uses commas as delimiters, as in the following example:

```
1,Allegra,Pseudois nayaur,5,824.8
2,Amela,Cerdocyon thous,11,840.8
3,Angela,Capra falconeri,8,387.2
5,Anjolie,Ailurus fulgens,10,941
6,Athena,Moschus fuscus,2,506.3
8,Ava,Cephalophus jentinki,13,135.5
```

Note that in this particular example, there are no animals with ids 4 or 7, as one of them passed away, while the other one was released back into its native habitat.

Input File
The binary file, `animals.dat`, is organized as follows: the first entry on the file is an integer that describes the number of available ids in the current database, followed by each animal's information, as shown in the structure defined above. All animals are listed in the increasing order by their id number, starting at value 1. If there is a hole

in id numbers, e.g. 4 and 7, then these structures' information is still present in the file, except a negative age value signifies an empty record.
In fact, the number of available ids in the input file is the number of holes in the id sequence, as in theory the overall number of ids is only limited by the amount of RAM we have on the system. The number of holes in the example given above would be 2.

Processing Logic

Searching for animals

The program is to search for one animal at a time and the search is to be based on the id number entered by the user. If an animal is found, and it is not an empty record, the program displays this animal's information to the screen. Otherwise, it prints an error message. In either case, the program returns to the main menu.

Note that searching needs to use random file access.

Deleting animals

Only one animal can be deleted at a time. Your program should give the user a choice to select an id number and verify it is a valid id. If the id is invalid, or it is an empty record, the program displays an error message and goes back to the main menu. Given a valid id, the program simply changes the animal's age in the file to a negative number and since the deletion creates a hole in the id sequence, the program increments the number of available ids in the file as well. After taking care of deletion, the program goes back to the main menu.

You should keep track of all deletion ids, as you can later on use those numbers for the insertion operation. You are allowed to use a data structure for that purpose.

Note that deletion itself needs to use random file access.

Adding animals

Only one animal can be added at a time and it is up to the program to select the animal's id. If there are holes in the id sequence, one of the holes needs to be used by the program. In the example given above, it would a value 4 or 7. If there are no holes in the id sequence, the program must append at the end of the database and pick the next consecutive number for the id, e.g. 9, if the database contains 8 animals numbered 1-7.

The program is to display the new id number and ask the user to enter name, species, age, and weight (in this order), and store this information in the file in the correct spot. Assume valid choice for these entries, except remember that species could consist of more than one word. Finally, if the hole was filled, the number of available ids in the file needs to be updated, and the program needs to return to the main menu.

Note that depending on the scenario, insertion needs to use either random or sequential file access.

Updating animals

Only one animal can be updated at a time. An update is a mix of search and animal insertion logic. First, the program asks for the animal's id number. If an animal is not found or it is an empty record, the program prints an error message and goes back to the main menu. Otherwise the animal information is displayed, and then the user is prompted to enter new name, species, age, and weight (in this order). The user is not given an option to update an id number. Assume valid choice for these entries, except remember that species could consist of more than one word. The program writes this information to the file in the appropriate spot and returns to the main menu.

Program Specs

- The code needs to be modular

- The minimum number of files for your program is 3 files
- You need to create a makefile for your code
- Name the executable file `animalbin`
- Your program has to follow basic stylistic features, such as proper indentation (use whitespaces, not tabs), meaningful variable names, etc.
- You need to comment your code
- You are not allowed to use global variables
- **Your program must compile in gcc gnu 90 – programs that do not compile will receive a grade of 0**
- Your program is to be compressed using tar and named <yournetid>_hw5.tar

Extra Credit (15%)

Open-ended – extend the program in some fashion – explain in comments at the top of your code.

Program Submission

On or before the due date, use the link posted in Canvas next to Homework 5 to submit your tar file. Make sure you know how to do that before the due date since late assignments will not be accepted.