Project 7, Program Design

You are given some data from an animal shelter, listing animals that they currently have. They have asked you to write a program to sort the dogs and cats in age in ascending order, respectively, and write them in separate files.

Assume the input file has the format of name (one word), species (one word), gender (one word), age (int), weight (double), with each animal on a separate line:

Hercules	cat	male	3	13.4
Toggle	dog	female	3	48
Buddy	lizard	male	2	0.3

….

Example input/output:

```
Enter the file name: animals.txt
Output file name:
sorted_dogs.txt
sorted_cats.txt
```

1. Name your program *animals.c*.
2. The output file name should be `sorted_dogs.txt` and `sorted_cats.txt`. Assume the input file name is no more than 100 characters.
3. The program should be built around an array of `animal` structures, with each `animal` containing information of name, species, gender, age, and weight. Assume that there are no more than 200 items in the file. Assume the name of an animal is no more than 100 characters.
4. Use fscanf and fprintf to read and write data.
5. Your program should include a sorting function so that it sorts the animals in age. You can use any sorting algorithms such as selection sort and insertion sort.

   ```
   void sort_animals(struct animal list[], int n);
   ```

6. Output files should be in the format of name gender age weight, with 2 decimal digits for weight. For example,

Toggle	female	3	48.01
Rocky	male	5	52.32

…..

Before you submit:

1. Compile with –Wall. Be sure it compiles on ***student cluster*** with no errors and no warnings.

gcc –Wall animals.c

2. Test your program with the script.

chmod +x try_animals

./try_animals

3. Submit both animals.c and animals.txt (for grading purposes).

Total points: 100

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality 80% (functions were declared and implemented as required)

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. Information to include in the comment for a function: name of the function, purpose of the function, meaning of each parameter, description of return value (if any), description of side effects (if any, such as modifying external variables)
4. Variable names and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.
5. Use consistent indentation to emphasize block structure.
6. Full line comments inside function bodies should conform to the indentation of the code where they appear.
7. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: `#define PI 3.141592`
8. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
9. Use underscores to make compound names easier to read: `tot_vol` or `total_volumn` is clearer than totalvolumn.