Leland Batey
CS 223
Homework 02 Requirements

Part 1 (75 points) :

Name:

- Wazoo

Requirements

Read a file that is a CSV that stores the information about animals. Each row has data about the animal. The first line/row is:

```
ANIMAL TYPE, NAME, AGE, FRIENDLINESS, WEIGHT, HUNGER
```

After the file is read, the program should accept the following commands on stdin:

1. EXIT
    - The program will exit gracefully, printing the following:
        - "THANK YOU COME AGAIN!
2. FEED N
    - N(as a floating poing number) is the number of food items to feed the animal.
    - The animals should be sorted by weight from lightest heaviest
    - Their hunger factor should be decremented by one (1)
    - Their weight should increase by 10% of weight * number of food items (.1wN)
        - E.G. a 100 lb animal fed 2 items should weigh 120 lbs after feeding.
3. PET
    - The animals should be sorted by friendliness from meanest to friendliest.
    - Their friendliness factor should be incremented by one.
4. SPEAK

- The animals should be printed to standard output in their current order.
- The animal data should be comma separated as shown below. This should math the input file format.
- You must prefix each animal with a saying based on their animal type.

```
bow-wow, DOG, FIDO,      15, 9,  185, 5
meow,    CAT, BOJANGLES, 1,  -4, 10,  50
moo,     COW, BESSY,     4,  1,  185, 5
```

Requirements and Notes:

1. Commands should be accepted case insensitive.
2. Animal types: dog, cat, cow
3. Animal speeches:
   - Dog = bow-wow
   - Cat = meow
   - Cow = moo
4. Each data column is separated by a column in the input file.
5. The friendlier the animal , the larger the value.
6. The hungrier the animal, the larger the value.
7. Weights must be positive. The program should exit with the following message if false:
   - `"Incorrect Weight Found"`
8. The animal's age must be positive. The program should exit with the following message if false:
   - `"Incorrcet Age Found"`
9. You can assume that the file will be correctly formatted (e.g. no text in numeric fields, no missing lines, no empty lines.)
10. Your program shold read and store the animals in a single container, e.g. a vector.
11. Your program should not display any other text, even when waiting for input.
12. You can use a vector to store the animals.

Example of executing program on the command line:

```
commandPrompt> wazoo animals.csv
```

Part 2 (25 points):

In the same program, **wazoo**, use *template classes* to implement a binary search tree. The search tree should be used to store each animal. When the user types "FIND" on the standard input, the progam should search for the animal by case-insensitive name:

```
FIND FIDO
```

If Fido exists, FIDO'S stats should be returned along with his pecking order suffixed to the output in CSV format:

```
DOG, FIDO, 15, 9, 185, 5, 10
```

Where 10 would be the 10th position in the feed or petting list. So, if FIDO is the meanest animal, his pecking order whould be 0 when the PET command issued.

If Fido does not exist, the following error message should be displayed:

```
"That animal does not exist in this zoo."
```

Duplicate animals:

> All animal names should be unique. If FIDO already exists once, you should rename the animals with roman numerals.
>
> DOG, FIDO II, 15, 9, 185, 5
>
> If there are four Fido's then the output would be as follows:
>
> DOG, FIDO IV, 15, 9, 185, 5

Requirements and Notes:

1. You may not use any STL or 3rd party tree libraries or objects. You must write your own BST.
2. You can assume that the roman numerals will go up to ten at most (e.g. `X`)

Grading Criteria:

> Inheritance
>
> Template Classes
>
> Correctness (includes not crashing and the output format meets the above criteria)

Submission:

1. Your submission must follow the syllabus rules.
2. You must submit your source code electronically
3. If the program does not compile, no credit is awarded. The homework will be tested on `elec.tricity.wsu.edu` and you should use the `g++` compiler.
4. You must turn in a hard copy of the assignment.
5. No late submissions.

Additional Questions:

1. Under both the feed and pet sections, there is a reference to "sorting the animals" by various metrics. Given that the internal "sort" structure might not matter, I assume this sorting is for presentational purposes. Am I correct that that happens as part of the `"print the animals by their current order"` under Speak?
    - This is correct. There must be an internal order maintained for use during printing.
2. Are we supposed to read the CSV over the stdin through redirection?
    - We'll be reading the CSV from within the program. We'll have the file name passed in on the command line.
3. If things are searched case-insensitively, does the case of the output (`DOG, FIDO IV, 15, 9, 185, 5`) matter? In fact, does the case of any of the output matter?
    - The printing is als case-insensitive.