CSC 1300 PROGRAM 2

Fall 2023

DINOSAUR FILE PARSING

IMPORTANT DATES:

Assign date: Wednesday, September 20, 2023

Due Date: Wednesday, October 18, 2023, 11:59pm

You may submit up to 3 days late at 10 points off per day late. This means you can't submit after October 21, 2023 11:59pm.

CONCEPTS:

- Branches & Loops
- String & Character Functions
- Programmer-Defined Functions
- Input & Output Files

Note: You may NOT use arrays, vectors, pointers, structures, or classes in this programming assignment.

FILES TO TURN IN:

You are given the three text files and they must be included in your submission. You must zip all the following files in a zip file and then upload the zip file to ilearn. **Points are deducted for not including all the files and submitting them in a zipped file**!

- **jdoe42_prog2.cpp** **(replace jdoe42 with <u>YOUR</u> TTU username!!!)**
- **dinoDirectory.txt** (provided for you)
- **carnivores.txt** (provided for you)
- **herbivores.txt** (provided for you)

DESCRIPTION / SPECIFICATIONS:

You are given a text file that has a data dump of dinosaur and other animal data. The format of the data in the file is as follows in this order:

1. Animal Name
2. Length and/or Width of animal
3. Mass/Weight of animal
4. What the animal eats
5. Description and facts about the animal

Each piece of information is separated by a '#'.

Your job is to read all the information from the file, determine if the animal is a dinosaur, herbivore, carnivore, or other type of animal. If it is an herbivore, you will print out the animal's information in a neat, easy to read format in an output text file named "herbOutFile.txt". If the animal is a dinosaur carnivore, you will print out the animal's information in a neat, easy to read format in an output file named "carnOutFile.txt". If the animal is not a dinosaur, then you will print it out in a text file named "otherOutFile.txt". Also, you will keep count of how many animals from the file are:

- Total dinosaurs
- Total carnivore dinosaurs
- Total herbivore dinosaurs
- Total dinosaurs who are over 10,000 pounds
- Total dinosaurs whose names contain the substring "saurus"
- Total animals that are not dinosaurs

To do this, you will create six accumulator variables and keep a running total when parsing through the dinosaur data file. The output of this program is short compared to the code required. You can see the sample output at the bottom of this document. Your output should match mine **<u>exactly</u>** (unless you change the dinosaurData.txt file).

REQUIRED FUNCTIONS

You may not have any global variables – even if they are constant. This is so you must practice sending data to/from functions!

Here are the function prototypes that you will need for your programmer-defined functions below.

You may NOT alter these function prototypes.

```
int carnOrHerb(string);
bool searchCarnivore(string);
bool searchHerbivore(string);
```

```
void printDino(ofstream&, ifstream&, string, int, int&);
bool overTenGrand(string);
```

Main Function

- Read in the filename of the data file from the user. (this is where the user will type **dinoDirectory.txt**)
  Note: Do not rename this file. When testing your program, we will type in this exact filename and will expect it will work.
- Open the file. Validate the file can be opened and if not, allow the user to enter in a different filename until the file they enter in can be opened. [Validate with **while** loop!]
- Start the loop to read from the file. I recommend your loop header be the following line
  `while(getline(dinoDirFile, tempString, '#'))`
  which will read the animal's name from the file.
  [Note: putting the '#' delimiter in while using getline will tell getline to read until it sees a '#' instead of just reading until it reaches a '\n', which is the default delimiter.]

- Inside this loop, you will do several things:
  - Does the name contain "saurus"? If so, increment the appropriate accumulator variable.
  - Call the **carnOrHerb** function, sending the animal name. This function will return an integer. If it returns a 1, then the animal is a carnivore. Otherwise, if the **carnOrHerb** function returns a 2, then the animal is an herbivore. Otherwise, the animal is "Other". Don't forget to increment the appropriate accumulator variables.
  - Depending on what the animal is, open the appropriate output file (**carnOutFile.txt** if it is a carnivore, **herbOutFile.txt** if it is an herbivore, or **otherOutFile.txt** if it is other).
    [Note: file must be opened in APPEND mode **(`ios::app`)** to work properly]

  - Then call the **printDino** function, sending the correct output file, the input file, the name of the animal, the appropriate accumulator for this animal, and then the accumulator for the total over 10,000 lbs. This function will be responsible for reading in the rest of the data about the animal from the input file, checking to see if the dinosaur is over 10,000 lbs, and printing the data in an easy-to-read format to the correct output file.

  - Don't forget to close the correct output file.

- After the loop completes that read in all the data and printed the data to the output files, then you can close the input file.

- Last, print the results to the screen. Refer to the sample output for the format and make sure your output matches the sample output **exactly**. This means you should use stream manipulators to create the columns. The lines above & below the output are made up of 50 dashes.

Make sure to check out the Sample Output Text Files given to you all in ilearn, which shows you what your carnOutFile.txt, herbOutFile.txt, and otherOutFile.txt should look like if your program works correctly.

carnOrHerb Function

Parameters: a string, holding the animal's name

Returns: an integer, indicating if the animal is a carnivore (1), herbivore (2), or other (-1)

This function should call the **searchCarnivore** function, sending the animal's name. This function will return a Boolean value, indicating if the animal is a carnivore (true) or not (false). If the searchCarnivore function returned true, you should return a 1 from this function.

If not a carnivore, the **searchHerbivore** function should be called, sending the animal's name. This function will return a Boolean value indicating if the animal is a herbivore (true) or not (false). If the **searchHerbivore** function returned true, you should return a 2 from this function.

If not an herbivore or carnivore, this function should return a -1.

searchCarnivore Function

Parameters: a string, holding the animal's name

Returns: a Boolean, indicating true if the animal is a carnivore and false if it is not

This function should open the text file named "**carnivores.txt**", which has a list of carnivore dinosaurs separated by a '#' delimiter. Once opened, you should read each dinosaur name with a while loop and check if the dinosaur name from the file matches the animal name sent to this function. If it does, then the animal is a carnivore and so you should return true.

If after checking every name from the file there are no matches, then this function should return false.

searchHerbivore Function

Parameters: a string, holding the animal's name

Returns: a Boolean, indicating true if the animal is an herbivore and false if it is not

This function should open the text file named "**herbivores.txt**", which has a list of carnivore dinosaurs separated by a '#' delimiter. Once opened, you should read each dinosaur name with a while loop and check if the dinosaur name from the file matches the animal name sent to this function. If it does, then the animal is an herbivore and so you should return true.

If after checking every name from the file there are no matches, then this function should return false.

printDino Function

Parameters: an output file stream passed by reference of the output file you are printing to, an input file stream passed by reference of the input file (**dinoDirectory.txt**), a string containing the name of the animal, the integer accumulator holding the running total of the kind of animal (carnivore, herbivore, or other), and the integer accumulator of the total number of dinosaurs over 10,000 lbs.

Returns: none

- Print the name to the output file.
- Read the height/length from the input file.
- Print the height/length to the output file.
- Read the mass from the input file.
- Call the **overTenGrand** function, which will return true if the dinosaur is over 10,000 lbs and if so, then increment the accumulator holding this total.
- Print the mass to the output file.
- Read what the animal eats from the input file
- Print what the animal eats to the output file.
- Read the description of the animal from the input file.
- Print the description to the output file.

overTenGrand Function

Parameters: a string, holding the mass of the animal

Returns: a Boolean, true if the animal is over 10,000 lbs and false otherwise

This function is going to challenge you in the problem-solving area (which is why this class is introduction to computer programming AND problem solving)! You may need to use a combination of string class functions and/or character functions to figure out if the animal could be over 10,000 lbs. The string sent to this function could be formatted like one of the following examples:

- `30 to 60 lbs`
- `5,000 to 9,000 lbs`
- `9,000 to 10,000 lbs`
- `11,000 to 15,000 lbs`

The first two examples above should return false and the last two examples should return true. Think about how you can extract out of this string just what you need (without any commas) in order to test the number. You will have to convert the number to an actual integer data type to do a proper comparison.

Some of these pages on cplusplus.com may help with this task:

- http://www.cplusplus.com/reference/string/string/
- http://www.cplusplus.com/reference/string/
- https://www.cplusplus.com/reference/cstdlib/

CODE READABILITY:

- Provide a comment block at the top of your source file that contains the source file name, your name, creation date, and purpose of the program. Put labels before this data like:
  Filename:
  Author:
  Creation Date:
  Purpose:

- Place a comment above each function telling the function name & purpose.

- Name your variables where they describe what they hold. Accumulator variables should have the word "total" or "sum" in them.

- Indent your code properly and consistently.

SAMPLE OUTPUT:

```
Dinosaur Directory File Name (dinoDirectory.txt):  dinoDirectory.txt

Tyrannosaurus Rex is being printed to the CARNIVORE file!
Ankylosaurus is being printed to the HERBIVORE file!
Stegosaurus is being printed to the HERBIVORE file!
Velociraptor is being printed to the CARNIVORE file!
Spinosaurus is being printed to the CARNIVORE file!
Triceratops is being printed to the HERBIVORE file!
Gerbil is being printed to the OTHER file!
Giganotosaurus is being printed to the CARNIVORE file!
Aye-Aye is being printed to the OTHER file!

--------------------------------------------------
```

```
                TOTAL DINOS:  7
     TOTAL CARNIVORE DINOS:  4
     TOTAL HERBIVORE DINOS:  3
     DINOS OVER 10,000 LBS:  5
DINO NAMES END IN 'SAURUS':  5
         ANIMALS NOT DINOS:  2
--------------------------------------------------
```