CPSC 122 Computer Science II

Gonzaga University

Daniel Olivares

PA2 – Pet Store 2
Search Algorithms, Pointers, and Dynamic Memory Management (100 points)

Individual, non-collaborative assignment

Learner Objectives

At the conclusion of this programming assignment, participants should be able to:

- Implement and use search algorithms to locate an item in a list of information (e.g., in arrays, vectors, and other data structures).
- Sort collections of values into order using sorting algorithms.
- Use and manipulate pointers and pointer operators.
- Allocate storage for variables using Dynamic Memory Management in C++.

Prerequisites

Before starting this programming assignment, participants should be familiar with C++ topics covered in CPSC 121 including (but not limited to):

- Conditional statements
- Implement loops to repeat a sequence of C++ statements
- Define/call functions
- Perform file I/O
- Work with 1D and 2D arrays
- Include and use the String and Vector libraries
- Demonstrate the software development method and top-down design

Overview and Requirements

This program will build upon the previous programming assignment to read information about the "inventory" of a pet store from an input file, `petstoredata.csv`, and output a summary report to an output file, `petreport.txt`. This program will require reading in and storing the entire contents of the pet store inventory into vectors and arrays using dynamic memory management, processing the data from those vectors and arrays, and then generating an output text report. Your program will also be required to make use of functions to process the pet data and compute portions of the summary report (some requiring search algorithms). Additionally, you are required to use pointers to indirectly access any numeric variables in your program.

Input File Format

The input file is in a CSV (comma-separated values) format. You are **not** allowed to modify the input file from the CSV format. The first line of the file contains the four (4) column header labels: Pet Store, Pet Name, Pet Type, Days at Store. You can assume that the input CSV file will ***-not-*** deviate from the 4-column format. Each following row contain data for each of the columns for a **variable number of pet store data (i.e., you do not know which pet stores are in the list or their data)**. You must be able to read and process **ANY number** of pets read from a file. That is, your program should work with any number of rows: 7 rows of data (1 row for the header, 7 rows of pet data) as in the example input file, 0 rows, 42 rows, etc.

Note: *when graded, your program will be tested with a different pet store input file that will follow the same format but will NOT have the same number of rows or pet stores as the example input file.*

Required Variables

This assignment has changed to require use of some specific variables. Additionally, I am also explicitly requiring that you need to make use of **parallel vectors and arrays** in your solution and are forbidden from using advanced data structures that we have not covered in the class yet (e.g., structs and classes). One change is the requirement for use of arrays (using dynamically allocated memory) for your integers (see **Gaddis Chapter 7.6 Focus on Software Engineering: Using Parallel Arrays**).

Reminder: **you are not allowed to use global variables for your solution**. All variables must be within scope of main() or your programmer-defined functions.

REQUIRED

```
int numDaysAtStoreSize = 0; //This variable keeps track of the size of your
                                        numDays dynamic array
int* numDaysAtStorePtr = nullptr; //This variable is a pointer to your dynamic
                                        integer array.
int uniquePetStoreNameCountsSize = 0; //This variable keeps track of the size of
                                        your unique counts dynamic array
int* uniquePetStoreNameCountsPtr = nullptr; //This variable is a pointer to your
                                        dynamic integer array
vector<string> petStoreNames; //Parallel Vector to petstore data vectors/arrays
vector<string> petNames; //Parallel Vector to petstore data vectors/arrays
vector<string> petTypes; //Parallel Vector to petstore data vectors/arrays
vector<string> uniquePetStoreNames; //Parallel Vector to unique count array
vector<string> alphabetizedPetNames; //Stores a copy of petNames alphabetized by
                                        the first character in the pet name
```

You are also **required to use a pointer to access any numeric variable**, e.g., if you declare an integer variable named `count`, you are required to also declare a pointer to that variable to access the variable indirectly, e.g., `int* countPtr = &count;` and then to increment count you would need to dereference countPtr for the operation, e.g. `(*countPtr)++;`

Suggestion/Tip: complete the program without using pointers for your numeric variables and then go back and change your program to use pointers.

Tip: Use "`uniquePetStoreNameCountsPtr`" in parallel with your "`uniquePetStoreNames`" vector, e.g., if pet store "`Fur Get Me Not`" is index 0 in your unique pet store name vector, index 0 in your unique pet store name count array storing a value of `3` means there are 3 instances of that pet store name in the input file.

	Value Stored in Data Structure	
Index	`vector<string> uniquePetStoreNames`	`int* uniquePetStoreNameCountsPtr`
0	Fur Get Me Not	3
1	Pick of the Litter	2
2	For Pet's Sake	2

SUGGESTED (here are a few suggested variables you may want to make use of)

```
vector<string> header;
vector<string> lineParts;
istringstream lineToParse; //Hint: you can make use of stringstreams to parse
                           strings based on a token - see the in-class example
double averageDaysAtStore = 0;
int daysAtStoreSum = 0;
bool firstRow = true;
bool matchFound = false;
int storeMostPetsIndex = 0;
int petOfTheMonthIndex = 0;
```

Required Functions

This assignment has changed to require use of some specific functions. *You are still required to create additional functions in order to demonstrate proper top-down design* (see **Gaddis Chapter 1.6 The Programming Process** and the **Software Development Method** sheet attached to this assignment) demonstrating modular programming with the use of functions (see **Gaddis Chapter 6 Functions**). Further, **I am explicitly forbidding use of external libraries that would solve parts of the problem for you** without you having to create an algorithm to solve it, e.g., you may not use a library for sets/maps, you must develop an algorithm to determine unique values etc. and use your vectors as part of the solution.

REQUIRED

- `int* pushBackInteger(int* originalArray, int* size, int newInt)`
  - Accepts an input array using pointer notation (NOT subscript notation), a pointer to an integer representing the size of the original array, and an integer value representing a new value to be added to the array.
  - Returns a pointer to dynamically allocated memory to an array one size larger than the original array including the new integer value at the back of the array.
    - e.g., original {42, 7} and newInt 9001 → returned is {42, 7, 9001}
  - Increments the size value to match the new size of the new array.
  - Hint: you cannot just increment size directly! Use pointer operators.
- `bool stringIsInVector(vector<string> searchVector, string targetString)`
  - Accepts a vector to search and a string to search for within that vector
  - Returns a Boolean value (i.e., this is a predicate function) indicating whether or not the target string was in the search vector.
  - Hint: this function should use a linear search algorithm.
  - Hint2: You should use this function to determine if a pet store name is unique.
- `void alphabetizePetNames(const vector<string> petNames, vector<string>& alphabetizedNames)`
  - Accepts a constant vector (i.e., read only) and an output vector (i.e. an output parameter or by reference).
  - Returns a copy of the petNames vector in **alphabetical order** by the **first character** of each pet name.
  - Hint: for this assignment you do not need to consider more than the first character, e.g., it wouldn't matter which order John and Jane are in the final vector as long as they are alphabetically ordered by the first character.
  - Hint2: `alphabetizedNames` can be an empty vector and you can make a copy of petNames into it to sort in place.
  - Design decision: you are free to select a sorting algorithm of your choice to implement

SUGGESTED

- Create one (1) function to open both of your input/output files.
    - I suggest you make it a predicate function which returns the success/failure status to the calling function.
    - Hint: you must use reference variables.
- Alternately, create two (2) functions for opening your files, one for your input file and one for your output file.
- Create one (1) function to read in **all** pet store information instead of having a loop read the data from main.
    - Hint: you must pass variables by reference (i.e., use **"output parameters"**). You cannot achieve this task with a return statement.
    - Hint: note the difference between passing a primitive data type (e.g., an integer) and an array by reference.
    - ***Tip:*** *if you are struggling with this function, complete the program by reading the input file in main first and then (after making a backup copy) try to move the segment of code into a function.*
- Create one (1) function to write all of your data to your output file.
    - Hint: all of your data should be stored in a number of vectors, you should not be reading from the input file and writing to the output file at the same time.
- Create functions for each of your calculations, e.g., one for calculating an average, one for determining which store has the most pets in the data set, one for randomly choosing a pet, etc.
    - Hint: recall – we strive for loosely cohesive and highly coupled functions, i.e., **one task, one algorithm, one function!**

Specifications

The input file pet information in the file consists of:

1. Store name (*string:* "`Fur Get Me Not`" is the name of the first pet in the example file below)
2. Pet name (*string:* "`Chris P. Bacon`" is the name of the first pet in the example file below)
3. Animal type (*string:* "`pig`" is the animal type of the first pet in the example file below)
4. Days in store location (*integer:* `1` is the number of days the first pet has been at this store in the example file below).
    - Days in store is guaranteed to be at least 1.
    - Pets are "moved to a different store" after 100 days. 😢

The **output** of your program has two parts:

1. Status messages displayed **to the console**. These messages simply let the user know what the program is currently computing (e.g., `Processed a pig, "Chris P. Bacon" ... 1 day(s) on site at store "Fur Get Me Not"`). *See the example console output below for the required format and output messages.*
2. Pet store summary information **written to an output file**. Name your output file `petreport.txt`. *See the example output file below for the required format and data*.
3. Note: THIS HAS CHANGED from PA1.

Main tasks:

1. Read all pet inventory information from `petstoredata.csv` for an **unknown number of pets** and output the processing information as it is read in from the input file.
    a. *Note that you cannot assume the input file contains a fixed number of records pets!*
    b. *You will need to use **parallel vectors/arrays** to keep track of each store name, pet name, pet type, and days at the store.*
    c. *Hint: read the input file line by line and then parse each line for the column data. I recommend using **getline** and **stringstream** to solve this problem (see in-class examples).*

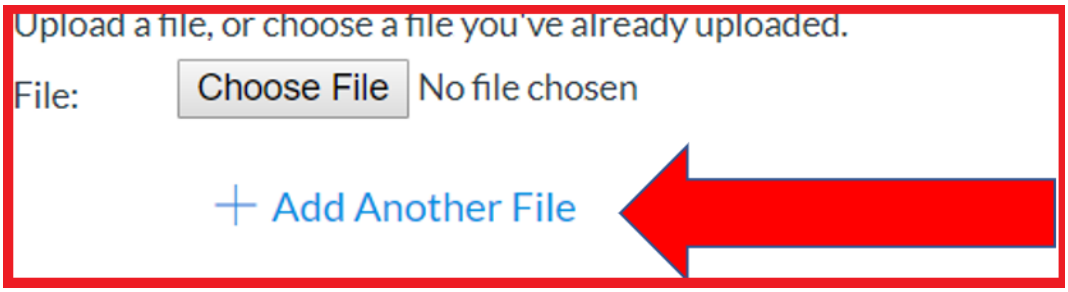
    d. ***Hint2***: you will need to use your required function, `pushBackInteger`, in order to add integer values to your dynamic arrays!
2. Keep track of the following information:
    a. Unique pet store names, e.g., in the example data we have **Fur Get Me Not, Pick of the Litter, For Pet's Sake**.
    b. Total number of pets in the input file, e.g., in the example data we have 7.
    c. Pet store with the most pets, e.g., in the example data this is Fur Get Me Not with 3 pets (the remaining have 2 each).
    d. Number of pets at the store with the most pets, e.g., in the example data Fur Get Me not has 3 pets.
    e. Pet average days on site across all stores, e.g., in the example data this should calculate to 31.
    f. Employee pet of the month choice. This should be **randomly chosen** from all pet data, e.g., in the example data there are 7 pets so you would randomly choose one of the 7 pets. If there were 100 pets you would choose one out of that 100.
    g. ***Hint:*** Create a **copy** of the petNames vector and alphabetize the new vector by the first character of each pet name.
3. Write the summary at the end of your output file.

Tip: I encourage you to use the software development method to plan out your algorithm to solve this problem and ***incrementally code-compile-test each portion of your program*** as you implement it! It may take a little longer at the start but I promise you that it will save you time and headache in the long run!

Note: **You should not need to use the string function** `stoi()`**,** `stod()` **or any variants for your solution. I am not going to forbid it but** you need to be able to understand and apply techniques used to overcome the issue encountered when switching between the extraction operator (>>) and the string `getline()` function.

Submitting Assignments

1. Submit your assignment to the Canvas course site. You will be able to upload your three source files (**two .cpp files and one .h file**) to the PA assignment found under the Assignments section on Canvas. You are ***REQUIRED*** to use the three-file format for this submission. Use the "+ Add Another File" link to allow choosing of three files (see reference image below).



2. **Your project must compile/build properly.** The most credit an assignment can receive if it does not build properly is 65% of the total points. Make sure you include all necessary libraries, e.g. string, ctime, etc. (though this does not mean add every single library you've ever used!)

3. Your submission should only contain your three source files (plus any bonus submission files). You may submit your solution as many times as you like. The most recent submission is the one that we will grade. *Remember to resubmit all three files if you submit more than once.*

4. Submit your assignment early and often! You are NOT penalized for multiple submissions! We will grade your latest submission unless you request a specific version grade (request must be made prior to grading).

5. If you are struggling with the three-file format, I recommend you complete the program in one file and submit that (working) version first, then convert it to the three-file format. **A working one-file format program is worth more points than a broken three-file format program!**

Note: *By submitting your code to be graded, you are stating that your submission does not violate the CPSC 122 Academic Integrity Policy outlined in the syllabus.*

Grading Guidelines

This assignment is worth 100 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 5 points for variable use and naming conventions
- 5 points for demonstration of top-down design and modular programming using functions
- 5 points for use of required vectors and random number generation
- 5 points for properly reading in csv data
- 15 pts for required variables
- 15 pts for required functions
- 15 pts for required use of numeric pointers to indirectly access numeric variable values
- 5 points for correctly calculating main tasks stats
- 5 points for correct console output
- 5 points for correct file output
- 5 points for using the three-file format and guard code
- 15 pts for adherence to proper programming style and comments established for the class

Example input file (`petstoredata.csv`)

```
Pet Store,Pet Name,Pet Type,Days at Store
Fur Get Me Not,Chris P. Bacon,pig,1
Pick of the Litter,Mary Puppins,dog,17
For Pet’s Sake,Jean-Clawed Van Damme,cat,60
Fur Get Me Not,Jack Meower,cat,24
Fur Get Me Not,Severus Snake,snake,12
For Pet’s Sake,Lame Duck,duck,5
Pick of the Litter,Barktholamew,dog,101
```

Expected Console Output

```
Processed 4 header columns: Pet Store, Pet Name, Pet Type, Days at Store

Processed a pig, "Chris P. Bacon" ... 1 day(s) on site at store "Fur Get Me Not"
Processed a dog, "Mary Puppins" ... 17 day(s) on site at store "Pick of the Litter"
Processed a cat, "Jean-Clawed Van Damme" ... 60 day(s) on site at store "For Pet's Sake"
Processed a cat, "Jack Meower" ... 24 day(s) on site at store "Fur Get Me Not"
Processed a snake, "Severus Snake" ... 12 day(s) on site at store "Fur Get Me Not"
Processed a duck, "Lame Duck" ... 5 day(s) on site at store "For Pet's Sake"
Processed a dog, "Barktholamew" ... 101 day(s) on site at store "Pick of the Litter"
All pet store data processed!

Alphabetizing pet names... done:
Barktholamew, Chris P. Bacon, Jean-Clawed Van Damme, Jack Meower, Lame Duck, Mary Puppins,
Severus Snake

Generating summary report...

Done!
```

Expected File Output (`petreport.txt`)

*NOTE: employee pet of the month choice is -**randomly chosen**- so your output will not always be the same on the last line.*

```
Pet Store CSV Summary Report

----------------------------


Pet Stores: Fur Get Me Not, Pick of the Litter, For Pet’s Sake

Total number of pets: 7


Pet store with the most pets: Fur Get Me Not

Number of pets at Fur Get Me Not: 3


Pet average days on site across all stores: 31

Employee pet of the month choice: "Chris P. Bacon"


Current Pet Inventory: Barktholamew, Chris P. Bacon, Jean-Clawed Van Damme, Jack
Meower, Lame Duck, Mary Puppins, Severus Snake
```