CSCI 1300 Introduction to Computer Programming
Instructor: Knox
Assignment 6
Due Sunday, October 22, by 6:00 pm
(5% bonus 6pm Fri Oct 20th, 2% bonus 6pm Sat Oct 21th)

This assignment requires you to write two files: *main.cpp* and *Assignment6.cpp*. There are four functions that you are required to write for this assignment. All these should be written in a file called *Assignment6.cpp*. If you write more functions than the those that are required, these should be kept in *Assignment6.cpp* as well. Your main.cpp file should just implement your **`main()`** function. Linking the two files can be done with **`#include "Assignment6.cpp"`** in your main file.

1. Once you have your code running on your virtual machine (VM), submit a zip file with *main.cpp* and *Assignment6.cpp* to the autograder COG .
   https://web-cog-csci1300.cs.colorado.edu

2. You **must also** submit your code to Moodle to get full credit for the assignment.
   - 10 points: Comments at the top of your source files should include your name, recitation TA, and the assignment and problem number.
   - 10 points: Algorithm descriptions comments in your code submission to describe what your code is doing.
   - **NEW!** 10 points: Code in your **`main()`** function that is testing the code in **`Assignment6.cpp`**.10 points will be deducted if the main.cpp you submit to moodle does not have code testing the functions you have written in **`Assignment6.cpp`**. Remember that COG *does not* run your `main()`, it runs its own main. (Code you write in `main()` will not impact your COG runs).

Assignment Details:

This problem is divided into four parts. Your main program should make a call to these four functions, passing the necessary parameters and getting the necessary return values. You **must** provide functions with the given names and parameters. The names of any other supporting functions are named at your discretion. The names of the supporting functions that you implement should be reflective of the function they perform.

You will not take any user inputs in the functions in your Assignment6.cpp *file. The problem statement is designed in a way that all the inputs to the functions are being passed from your main function. Secondly, you will* ***not*** *be printing any values in these functions. You need to evaluate the correctness of your implementation by writing code in your main program and validating the return values from these function. If you add debugging outputs, please remove that code before submitting.*

Treat every function as an independent function, where each function can take a different array as an input. Each of these functions will return a value or modify an array parameter. Your main should be used to setup the parameters and print out the returned value(s) of the function.

Homework Problems:

There are four parts to this homework. The following is the step by step description of each.

Part 1 – Read a file line by line

In this part, you will read in a file line by line.

```
float avgCharsPerLine(string filename)
```

The **avgCharsPerLine** function will take one argument, a string for the name of the file to be read in. Read the file line by line and return a floating point value for the average number of characters on each line. *Include every single character* (including white spaces) in the character count.

Part 2 - Read floats into 2 dimensional array

In this part, you will read a file with floating point values separated by commas and store them in an array.

```
int fillArray(string filename, float array[][5])
```

The **fillArray** function takes in two arguments: the filename and a float two dimensional array that can hold 5 floats in each row. Your function should fill the array with the values from the file, and return the total number of lines of the file, excluding the header line.

The input file will look like this. The *first line is a header line that describes the columns of data in the file. This line must be ignored when reading the data*.

```
Assignment1, A2, A3, A4, A5
90.2, 80, 0, 75.4, 98.2
94, 93.5, 92, 88, 87.5
80.2, 76, 88.2, 90.1, 82
```

For the above input, you should get an array like this:

90.2	80	0	75.4	98.2
94	93.5	92	88	87.5
80.2	76	88.2	90.1	82

HINT: Think about how you could use the `Split` function you wrote in recitation for parsing lines.

Part 3 – Calculate stats for two dimensional array

In this part, you will be working with a two dimensional array.

```
float arrayStats(string filename, float numbers[][5])
```

The **`arrayStats`** function takes in two arguments: a filename and a 2D array of floats that can hold 5 floats in each row. The input file will have the same format as the file in Part 2. You can call **`fillArray`** in this function.

The input file will have a variable number of rows but exactly 5 floats on each row, and a header line.

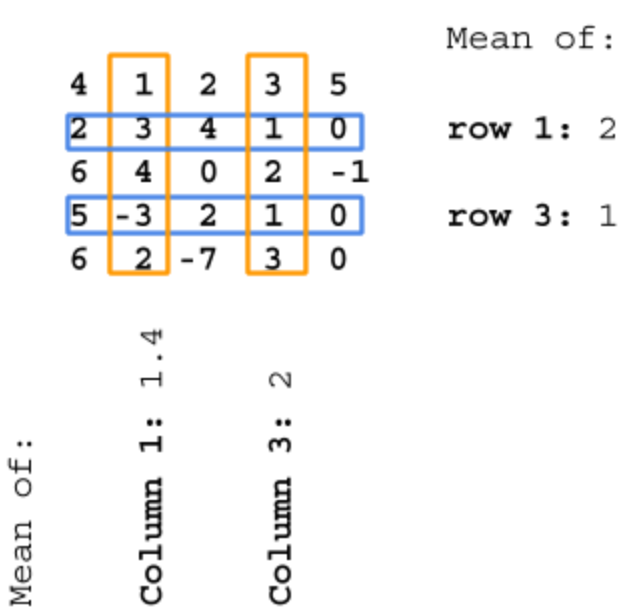
Format of Input File

```
value1, value2, value3, value4, value5
4, 1, 2, 3.8, 5
2, 3.2, 4, 1, 0
6, 4, 0, 2.5, -1
5, -3, 2, 1, 0
```

Read in the values from the input file, then calculate the mean of every **odd row** and **odd column** (row/column 1, 3 ,…) and sum these values. Return the sum as a float.

For this example, the sum of the means of the odd columns is `3.4` and the sum of the means of the odd rows is `3`. So the function **`arrayStats`** should return `6.4`.

WARNING: Make sure to do the calculations by looping over *individual* columns and rows. Values calculated using other methods may result in discrepancies with expected output in COG due to rounding.



Part 4 – Book ratings by users in two arrays

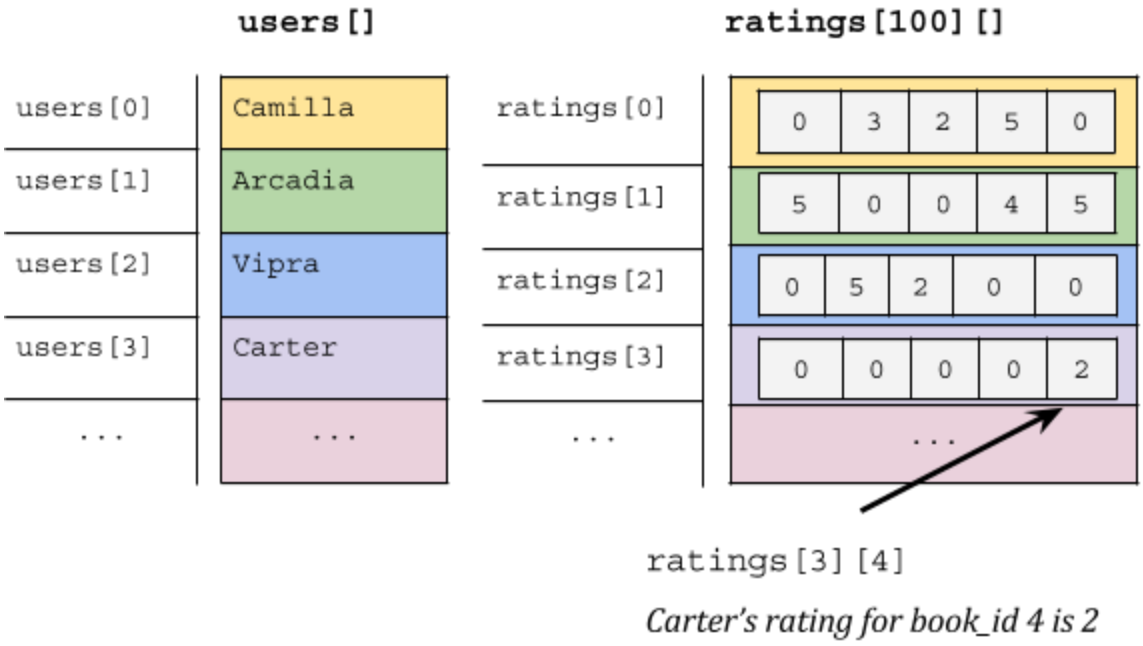
Let's say we are running a small online bookstore. We'd like to eventually create a book recommender system, so we've decided to start by storing the ratings that our users have provided for our books in arrays.

Our rating data is currently in text files. The first line of an input file is a header and should be ignored. Every other line represents a rating that a user has given to a book. Each line has user's **`name`**, the **`book_id`** that identifies the book they are rating, and their **`rating`**.

Notice that `Camilla` is on both the first line and the last line. The first line has her rating for the book with `book_id` 2. The last line has her rating for the book with `book_id` 24. One user can rate multiple books, but cannot rate the same book twice.

Format of Input File

```
Name, book_id, rating
Camilla, 2, 5
Arcadia, 3, 4
Vipra, 4, 5
Carter, 28, 2
Carter, 29, 4
Tom, 24, 3
Camilla, 24, 3
```



For this question, you will be filling *two* arrays. One will be an array of users, and the other will be a two dimensional array of book ratings for each user. Each unique user in the

input file should have one entry in **users** and one row in **ratings**.

New users should be added to the arrays in the order they are first encountered in the input file. For the example input file we provided, the users array should be in the order presented in the above figure.

The array at **ratings[i]** represents *all* the ratings that a particular **users[i]** has left. The value at **ratings[i][j]** is **user[i]**'s rating for a particular book **j**. Book ratings are integer values in the range [1, 5]. The rating 0 indicates that the user has not rated the book. There will be a maximum of 100 users and a maximum of 50 book ids (the range of book_ids is [0, 49].

```
void addBookRatings(string filename, string users[], int ratings[][50])
```

Write the function **addBookRatings.** It should take three arguments: a filename, an array of strings, and a 2-D array of integers. The function does not return any values, but should modify the elements of the array **users** and array **ratings** using the data provided in the input file named **filename.**

Hints:

Think about the task of processing *one* new line in the file:

```
Camilla, 2, 5
```

Is the name on this line is already in the **users** array? (How would we check?) If it is, we can add the rating information to the appropriate position in the **ratings** array. Otherwise we'll need to add a new user to the users array first.

What should happen when a new user is added to the **users** array? They will need a corresponding row in the **ratings** array. (Remember that any books that haven't been read should be marked by a 0 in the user's ratings arrays!)

Super Challenge Problem (3d array, parsing and reading from file)

When we study a physical system we need to store many data points to have a better representation of the object in space. Thus to describe that specific object we need more than one variable. In euclidian space we can define a set of coordinates to describe where an object is. We could also keep track of the velocity of the object at that specific location in space. It follows that a set of variables for an object O could be {x, y, v} where x = x position, y = y position, v = velocity.

Let's assume that we are mad physicists and we need to keep track of the identity of the object (specified by an ID), its x and y position, and velocity. Write a function that stores all

the information from a given text file of the following format in a 3d array (an array of arrays of arrays!). Every line will be in the format shown below, but there will be a variable number of lines.

I should be able to obtain the *velocity* of some object by accessing my array with the id, x, and y values in that order.

Format of Input File

```
id, x, y, velocity
11, 0, 1, 3
5, 1, 7, -9
6, 30, 27, 16
105, 300, 44, -19
```

I should be able to get the velocity of the object with id=11 at x=0 and y=1 by doing something like this:

```
Objects[11][0][1]
```