CSCE 121 Exam 2

Version 20.10.28-30

# Academic Integrity

Remember:

* Aggies do not lie, cheat, or steal, nor tolerate those who do.
* I know that you know a lot of things and your computational thinking is evolving.
  + I hope you have fun solving these problems, even forgetting this is an exam ☺
* You can do this!

# Instructions

1. Download the starter code from Mimir
   * Do not change the file names. Otherwise, it will not be graded.
   * Filenames
     + weatherFunctions.h
     + weatherFunctions.cpp
     + weather.cpp
       - We won’t actually grade this, but you can use main to debug.
2. Get started
3. Submit early, submit often on Mimir.
4. Work alone
5. Ask for clarification on Piazza (private posts only)

# Problem Overview

You are writing a program that analyzes rainfall data collected from weather stations in different cities over a range of days.

* You will load and analyze rainfall data for different cities for a period of days.
  + Examples of rainfall on a day are
    - 0 inches
    - 0.5 inches
    - 2.8 inches
    - 1 inch
    - 2023 inches
      * Dr. Ritchey’s deluge...
* You will read the data from a file into a table (i.e. a **two-dimensional array**).
  + Each **row corresponds to a city.**
  + Each **column corresponds to a day.**
  + The element in the 2D array at the row r and column c denotes the amount of rain in the city r on day c.
  + Your 2D array must be structured as we have done in this course. There are other approaches that you might find online, but they are not guaranteed to work with our test cases.
* You will write functions that analyze the data to find averages and maximum values.

To support automated grading of your program, as you implement functions

* You must maintain the ordering of the parameters presented.
  + You must decide the correct/appropriate data type to use for each parameter.
* You must keep the exact function name.
* You must ensure that the function signature (i.e. parameters) results in the function operating as indicated.
* Use data types that match the description.
  + If you need a floating-point number, use double.
  + If you need a signed integer, use int.
  + If you need an index or a size, use size\_t or unsigned long ~~or unsigned int~~.
    - Do not use unsigned int.
  + If you need a string, use the C++ string class.

**Allowed Includes: use of an “illegal” header file will result in a zero (0) on the exam.**

* iostream
* fstream
* string
* stdexcept
* weatherFunctions.h

**You may assume that the data/input has no errors unless indicated otherwise.**

**It is recommended to use the following compiler flags when testing locally:**

g++ -std=c++17 -Wall -Wextra -pedantic -g

**Your code must compile without error. Submissions which do not compile will receive zero (0) credit.**

**To check for memory leaks,**

* **add the Address Sanitization flag and run:**

g++ -std=c++17 -Wall -Wextra -pedantic -g **-fsanitize=address,undefined**

./a.out

* **and/or run Valgrind:**

g++ -std=c++17 -Wall -Wextra -pedantic -g

valgrind --leak-check=full ./a.out

# Example

This is an example that illustrates the use of the functions you will write. See the [detailed function requirements](#_8ne7jeregfec) after the example.

Start by assigning

A variable which stores the **table** to the null pointer (nullptr)

A variable which stores the **number of cities** to 0

A variable which stores the **number of days** to 0

Given a file containing the following data (cities1.txt):

2

3

2.0 0 0.4

0 3.2 8

Pass the variables which store the **table** (which is currently nullptr), **number of cities**, **number of days**, and the **name of the file** (cities1.txt) into [add\_cities\_from\_file](#_l60m6s9v11iu) which returns a pointer to a table

Table

|  |  |  |
| --- | --- | --- |
| 2.0 | 0 | 0.4 |
| 0 | 3.2 | 8 |

With Other changes

* **Number of cities** - 2
* **Number of days** - 3

Assign the returned value to the variable which stores the **table**

Given a file containing the following data (cities2.txt):

3

3

1.2 0.25 0.3

0 0 0.1

2.3 2.9 4.25

Pass the variables which store the **table** (which currently has 2 rows and 3 columns), **number of cities** (currently 2), **number of days** (currently 3), and the **name of the file** (cities.2.txt) into [add\_cities\_from\_file](#_l60m6s9v11iu) which returns a pointer to a table

Table

|  |  |  |
| --- | --- | --- |
| 2.0 | 0 | 0.4 |
| 0 | 3.2 | 8 |
| 1.2 | 0.25 | 0.3 |
| 0 | 0 | 0.1 |
| 2.3 | 2.9 | 4.25 |

With Other changes

* **Number of cities** - 5
* **Number of days** - 3

Using this **table** for the following:

* Calling [get\_average\_rainfall\_between\_days\_given\_city](#_j8m4wgiobtze) with start day 1, end day 2, and city index 1 returns 5.6.
* Calling [get\_average\_rainfall\_between\_days\_all\_cities](#_ve0u4chit1lh) with start day 0 and end day 1 returns 1.185.
* Calling [get\_max\_rainfall\_all\_cities\_given\_day](#_8jpdlf8vtxv0) with day 1 returns 3.2.
* Calling [get\_max\_rainfall\_given\_city](#_v97ajgq53hao) with city index 4 returns 4.25.

Calling [destroy\_table](#_p0oahqi5y4cm) with arguments of the **table** and the **number of cities** (5) results in all of the memory for the table being freed/destroyed/deleted/deallocated from the heap.

# Required Functions

You are required to provide these functions, but you may add additional helper functions that support the required functions.

## 

## add\_cities\_from\_file

### Input Parameters

* Pointer to a table (i.e. a two-dimensional array)
* number of cities
* number of days
* the filename (as a string) of a file with data in the following format
  + You can assume the data is valid.

### File Format

<number of cities as a positive integer>  
<number of days of rainfall data as a positive integer>  
<a space-separated list of floating-point rainfall for city 1>  
…  
<a space-separated list of floating-point rainfall for city n>

### Return

* pointer to a table (i.e. two-dimensional array) on the heap with the new data added
  + If the input table is the null pointer
    - create a new table
  + Else
    - resize the table and add the new data
      * Remember that resizing includes deallocating the old array

### Other changes

* updates number of cities
* updates number of days

### Throws

* std::invalid\_argument (hereafter “invalid argument”)
  + if unable to open the file
  + if the number of days in the input file is not equal to the number of days in the existing table

## 

## destroy\_table

### Input Parameters

* pointer to a table (i.e. two-dimensional array)
* number of cities

### Return

* None (void)

### Other changes

* Deallocates table from the heap

### Throws

* Invalid argument
  + If number of cities is zero

## 

## get\_average\_rainfall\_between\_days\_given\_city

### Input Parameters

* pointer to a table (i.e. two-dimensional array)
* number of cities
* number of days
* index of start day
* index of end day
* index of city for which to find the average rainfall

### Return

* Average rainfall for given city from start day to end day (inclusive)

### Other changes

* none

### Throws

* invalid argument
  + Invalid table (table is nullptr)
  + Invalid index of city (index >= number of cities)
  + Invalid Start Day index (index >= number of days)
  + Invalid End Day index (index >= number of days or index < index of start day)

## 

## get\_average\_rainfall\_between\_days\_all\_cities

### Input Parameters

* pointer to a table (i.e. two-dimensional array)
* number of cities
* number of days
* index of start day
* index of end day

### Return

* Average rainfall for all cities between start day and end day (inclusive)

### Other changes

* none

### Throws

* invalid argument
  + Invalid table (table is nullptr)
  + Invalid Start Day index (index >= number of days)
  + Invalid End Day index (index >= number of days or index < index of start day)

## 

## get\_max\_rainfall\_all\_cities\_given\_day

### Input Parameters

* pointer to a table (i.e. two-dimensional array)
* number of cities
* number of days
* index of the day for which to find the maximum rainfall

### Return

* Maximum rainfall for all cities for the given day

### Other changes

* none

### Throws

* invalid argument
  + Invalid table (table is nullptr)
  + Invalid number of cities (zero cities)
  + Invalid Day index (index >= number of days)

## 

## get\_max\_rainfall\_given\_city

### Input Parameters

* pointer to a table (i.e. two-dimensional array)
* number of cities
* number of days
* index of the city for which to find the maximum rainfall

### Return

* Maximum rainfall for the specified city for all days

### Other changes

* none

### Throws

* invalid argument
  + Invalid table (table is nullptr)
  + Invalid number of days (zero days)
  + Invalid index of city (index >= number of cities)