**CSE-381: Systems 2**

**Homework #1**

**Due: Wednesday September 4 2019 before 11:59 PM**

**Email-based help Cutoff: 5:00 PM on Tuesday Sept 3 2019**

Maximum Points: 23

|  |
| --- |
| **Submission Instructions**  This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source code is named *MUid\_homework1*.cpp (where MUid is your Miami Unique ID, e.g., raodm\_homeweork1.cpp). Ensure your program compiles (without any warnings or style errors) successfully. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload just the 1 source file onto Canvas via the CODE plug-in.   1. The 1 C++ source file developed for this part of the homework.   **General Note**: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc. |

|  |
| --- |
| **Objective** |
| The objective of this homework is to refresh your memory on the following concepts and skills from “[CSE-278: Systems 1](http://miamioh.edu/cec/academics/departments/cse/academics/course-descriptions/cse-278/index.html)” prerequisite course:   * Basics of HTTP * Working with sockets * Basic I/O operations in C++ * Working with unordered\_map. * Developing C++ program involving simple string manipulation. * Working with command-line arguments. |

# Grading Rubric:

|  |  |
| --- | --- |
|  | The program submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet base case requirements or just skeleton code will be assigned zero score!  Program that do not compile, **have even 1 method longer than 25 lines**, or just some skeleton code will be assigned zero score. |

* **-1 Points**: for each warning generated by the compiler.
* **NOTE:** Violating CSE programming style guidelines is a compiler error! Your program should not have any style violations reported in NetBeans when you compile it.
* **Base case:** 13 points
* **Additional feature:** 7 points
* **Formatting & Documentation:** 3 points (if your comments are poor, then you loose points int his category)

# Starter Code:

Use the starter code from Exercise #1 for this homework. A couple of additional headers you will need are: <unordered\_map> and <iomanip> (for std::quoted method)

# Background

In this homework you will be developing a custom HTTP client to dynamically retrieve sales data via a given URL and print total sales by a given employee. Optionally (extra feature) your program also prints the employee with highest total sales. The inputs will be supplied as command-line arguments to the program (see order of arguments further below).

Note the following aspects:

1. The URL supplied to your program will always refer to a valid file, examples: <http://ceclnx01.cec.miamioh.edu/~raodm/cse381_fall19/homework/hw1/sales_2017.tsv>, <http://pc2lab.cec.miamioh.edu/courses/sales_2018.tsv>, <http://www.users.miamioh.edu/raodm/courses/sales_2019.tsv>
2. Each file contains 2-or-more lines. **Note**: The first line of each file is a header line. Read and ignore this line prior to processing entries.
3. Each line has sales data stored in the following order (with 1 tab-character separating each column): EmployeeName\tCustomerInfo\tSales. The sales price is rounded to nearest dollar figure and hence is always an integer.

# Homework Exercise

In this homework you are required to develop a C++ program to process sales data that is dynamically downloaded from a given URL. The inputs to the program will be supplied as command-line arguments in the following format:

|  |
| --- |
| $ ./homework1 “url” “employee” MaxFlag |

Where,

* 1st command-line argument (*i.e.*, argv[1]): Will always be a valid URL that contains a valid sales data text file for the program to process
* 2nd command-line argument (*i.e.*, argv[2]): Will always be name of an employee. However, the employee name may or may-not be present in the sales data file and your program should handle these 2 cases appropriately as shown in sample outputs.
* 3rd command-line argument (*i.e.*, argv[3]) MaxFlag: This command-line argument will be a string with value of either “true” or “false”. If this argument is “true” then the program should print the employee with maximum total-sales as shown in sample outputs further below.

**Base case**: In order to earn any points at all for this homework, the program must correctly operate for the following functionality –

* If the specified “employee” (2nd command-line argument) entry is found in the given sales data, then the total-sales for the employee should be printed. For example, in the sales data <http://pc2lab.cec.miamioh.edu/courses/sales_2018.tsv>, if you add up the sales entries for “Bob Smith”, the total sales for this employee is 12784 and the program should print this value. See sample outputs for format.
* If the specified “employee” was not found in the sales data then print a message “Employee Bob Smith not found.” See sample outputs for format.

**Additional Functionality**: If the “MaxFlag” (3rd command-line argument) is true, then print information about the employee with maximum total-sales. For example, in the sales data <http://pc2lab.cec.miamioh.edu/courses/sales_2018.tsv>, employee “Mary Jane Lee” has the highest total-sales of 42057. See sample outputs for format.

## Tips:

1. Review and suitably use parts of code from exercise1.cpp (i.e., CSE-278)
2. Review concepts of command-line arguments already covered in CSE-278. Watch the video on setting command-line arguments in NetBeans on [Canvas →Pages→Video Demonstrations](https://miamioh.instructure.com/courses/103114/pages/video-demonstrations). Here is a screenshot of an example of command-line argument settings for this homework in NetBeans:



1. Rest assured that I/O is trivial using std::quoted method. Reading data is almost just 1-line of code, if done correctly.
2. It would be easiest to aggregate total-sales for each employee in an std::unordered\_map with employee’s name being the key and the sales figure being the value stored in the map.
3. The find method in unordered\_map will come in handy.
4. Review slides on how to use a simple range-based for-loop to iterate through an unordered\_map.
5. First write skeleton code with plenty of comments to help you work out the solution.
6. **Save and compile often – you should type no more than 3-or-4 lines before saving and compiling.**
7. For troubleshooting use the debugger. There is video demonstrating the use of the debugger in NetBeans [Canvas →Pages→Video Demonstrations](https://miamioh.instructure.com/courses/103114/pages/video-demonstrations). Ensure you review the video on how to use the debugger.

## Sample inputs and outputs

Note: Commands are all typed on 1 line but appear wrapped in the output below. User inputs are shown in **bold**

**Base case #1:**

|  |
| --- |
| **$ ./homework1 "http://pc2lab.cec.miamioh.edu/courses/sales\_2018.tsv" "James Smith" false**  Sales by James Smith = 13900 |

**Base case #2:**

|  |
| --- |
| **$ ./homework1 "http://www.users.miamioh.edu/raodm/courses/sales\_2019.tsv" "Ron Mc\"Gee" false**  Sales by Ron Mc"Gee = 54134 |

**Base case #3:**

|  |
| --- |
| **$ ./homework1 "http://ceclnx01.cec.miamioh.edu/~raodm/cse381\_fall19/homework/hw1/sales\_2017.tsv" "Ben Smith" false**  Employee Ben Smith not found. |

**Additional functionality Test #4:**

|  |
| --- |
| $ **./homework1 "http://pc2lab.cec.miamioh.edu/courses/sales\_2018.tsv" "James Smith" true**  Sales by James Smith = 13900  Top employee: Mary Jane Lee with sales: 42057 |

**Additional functionality Test #5:**

|  |
| --- |
| **$ ./homework1 "http://www.users.miamioh.edu/raodm/courses/sales\_2019.tsv" "Ron Mc\"Gee" true**  Sales by Ron Mc"Gee = 54134  Top employee: Ron Mc"Gee with sales: 54134 |

# Turn-in:

This homework assignment must be turned-in electronically via Canvas CODE plug-in. Ensure your C++ source code is named *MUid\_homework1*.cpp (where MUid is your Miami Unique ID). Ensure your program compiles (without any warnings or style errors) successfully. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

* The 1 C++ source files you developed as part of this homework.

Upload all the necessary C++ source files to onto Canvas via the CODE plug-in. Do not submit zip/7zip/tar/gzip files. Upload each file independently.