CSE 465/565
Homework #4: (50) points

Outcomes:

- Working with C# as one of Imperative Programming Languages
- Understanding how C# works and some of the difficulties
- Working with some of C# features
- Working with files in C#
- Solving problems using C#
- Running and testing C# codes

Scoring (465/565):

- (5 points) submission on Git with **10 or more commits**. Your Git history should represent the progress of your work.
- (17 points) Using all of the following C# features:
    - (8 pts) Use of Object-Oriented Programming :
        - Design a logical and practical hierarchy of classes/structs with at least one superclass and a subclass (3 pts).
        - Using setter and getter methods (2 pts).
        - Override an operator, which could be either a logical operator, or a mathematical operator (2 pts).
        - Override a method (1 pt).
    - (1 pt) Aliases.
    - (1 pt) Matrix Array.
    - (1 pt) Data Structures
    - (2 pts) Delegate.
    - (2 pts) Nullable Variable.
    - (2 pts) Different types of parameter variables.
- (24 points) Successful implementation of 3(undergraduate)/4(graduate) problems.
- (4 points) The execution time and style of your code including indentation, comments, etc.
- (5 bonus points) Pointers are used effectively.

Requirements:

- On your laptop, add a new folder inside your **CSE465_565** folder, and call it **Homework4**.
- Download **Hw4.cs**, **zipcodes.txt**, **zips.txt**, **cities.txt**, and **states.txt** from canvas and put them inside the **Homework4** folder.

Instructions:

1. In your project, you are free to create as many classes within the **Hw4.cs** file or across multiple files as you need. However, ensure that the **Hw4.cs** file is the only one that contains a **Main** method. This method should be within a class named **hw4**. This specific setup is crucial because your instructor will use the **hw4** class to execute and evaluate your work.
2. The programs will process the datafile **zipcodes.txt** which contains over 81K lines of text, one line for each geographic location in the US territories.
    - The first line of the file is the header line that explains each column of the data.
    - Except the first line, all other lines are tab-separated.
3. When **Hw4.cs** is run, it should create the following output files:
    - **CommonCityNames.txt**: contains all the common city names that appear in all of the states listed in **states.txt**.
        i. Each city in the file must be unique, no duplicate names allowed.
        ii. The cities should appear in sorted order.
        iii. The cities should appear one per line.
    - **LatLon.txt**: for each zip code listed in **zips.txt**, there will be a corresponding line of output. Each output line will list the zip code's latitude and longitude.
        i. The latitude and longitude must be separated by a space on each line.
        ii. If a zip code has multiple entries, provide the first one listed in zipcodes.txt.
    - **CityStates.txt**: For each city listed in **cities.txt**, there will be a corresponding line of output. Each output line will list the states containing that city name.
        i. The states must be separated by a space.
        ii. The states should appear in sorted order.
        iii. Each state should be listed only once for each city.
    - **grad.txt**: **(CSE565 only)** Display each of the following pieces of information - one per line
        i. The two geographically most distant zipcodes - space separated
        ii. The zip code with the largest population
        iii. The zip code with the smallest population
        iv. The zip code with the largest per capita wages
        v. The zip code with the smallest per capita wages
        vi. The city with the largest total population (e.g., Brooklyn, NY).
4. In order to get full points you must use features discussed in the class as following:
    - Use of Object-Oriented Programming :
        i. Design a logical and practical hierarchy of classes/structs with at least one superclass and a subclass.
        ii. Using setter and getter methods.
        iii. Override an operator, which could be either a logical operator, or a mathematical operator.
        iv. Override a method.

    - Aliases.
    - Matrix Array.
    - Proper Data Structures.
    - Delegate.
    - Nullable Variable.
    - Different types of parameter variables.

5. Clarity and correctness is the primary consideration when writing your codes.
6. Performance is the secondary concern; however, using obviously wasteful and brute force approaches will result in a loss of points.
7. You can compile and execute your code via Command Line (CMD)/Terminal using the following commands:
    - mcs *.cs (compile)
    - mono Hw4.exe (execute)
8. Commit and push your work right after any small progress. Your Git should demonstrate the progression of your work.
    - Adding or removing empty lines or spaces in one commit, does not add authenticity to your work.
    - Committing and pushing large volumes of code within a very short time frame, such as minutes or seconds apart, can raise doubts about the work's originality. Producing significant amounts of code this quickly is unrealistic and may indicate that the code is copied from another source. Genuine coding usually requires more development time and deliberate, thoughtful submission of changes.
9. The execution time and style of your code, including indentation and comments, should be acceptable. Brute force approaches which lead to very slow programs, and a lack of clarity will result in a loss of points.

Bonus Points:

To earn 5 bonus points on your assignment, you need to use pointers effectively in your program. Additionally, you must clearly indicate where you have used these pointers in your code. Here's how you do it:

- Use pointers in your code where it makes sense and helps with the performance of your code.
- At the very beginning of your **Hw4.cs** file, add special comments (the format is explained below) to point out exactly where in your code you've used pointers.

For example, if you used pointers between lines 10 and 15, and again between lines 40 and 63, you should add these lines at the top of your **Hw4.cs** file:

// => Used Pointers from lines 10 **to** 15 **<=**
// => Used Pointers from lines 40 **to** 63 **<=**

Replace "10" and "15" with the starting and ending lines of the first section where you used pointers, and "40" and "63" with the starting and ending lines of the second section where pointers are utilized.

If there is only one section or if there are additional sections where pointers are used, add a single comment for the former or continue listing them in the specified format for the latter.

- Failing to highlight the use of pointers by not adding the required comments at the top of your file will result in 0 bonus points.

By default, these 5 bonus points will be added to your overall Homework grade. However, you have the option to request that these points be applied to either your Quizzes or Exams instead. Please note, these points cannot be applied to Attendance. To make this request, simply email your instructor specifying where you would like your bonus points to be added.

Test your program:

Come up with more tests and test your program comprehensively. Your code will be tested separately with different text files.

Proper testing requires time. Begin by considering various scenarios and delving into the specifics. Eventually, you'll need to come up with a way to validate your solutions using the data at hand. Remember, thorough testing demands quality time devoted at the end to ensure your code functions as intended. So, as you manage your schedule for this assignment, make sure to allocate sufficient time for testing. The consequence of lack of proper testing will result in losing points.

Submission:

Submit the GitHub/GitLab URL of the project **Homework4**.
Inside this folder there should be the following files:

1. **Hw4.cs** and other complimentary classes.
2. All the input and output text files.