Homework #05

Complete By: Part 1: noon on Wednesday April 14th
Part 2: noon on Friday, April 16th
Part 3: noon on Monday, April 19th

Policy: Individual work only, earlier parts may submit by final deadline at 10% overall penalty

Assignment: Fill in the commented out section in C# code

Submission: submit electronically on Gradescope

Resources

Some resources to help you understand the basics of C# and help you in your project:

A general introduction to C# provided by Microsoft.
https://docs.microsoft.com/en-us/dotnet/csharp/
An interactive C# tutorial (may be very basic coming from C++ experience)
https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/tutorials/hello-world
A series of introductory videos – use the lectures from this class as a primary resource, the topics tend to wander
https://channel9.msdn.com/Series/CSharp-101/?WT.mc_id=Educationalcsharp-c9-scottha
C# features by topic
https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
More information on the type system and value vs reference types
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/
Quick comparison of how to use common constructs between languages
http://www.programmersguidetothegalaxy.com/#
Code snippet showing how functions are called in C#. Has good small code snippets organized according to specific topics to help understand basic concepts of C# well.
http://github.com/dotnet/training-tutorials/blob/master/content/csharp/getting-started/README.md
Examples of how to pass parameters by value and reference into functions.
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/passing-parameters
A more detailed example with explanation on passing parameters by reference (ref) into functions.
https://github.com/dotnet/docs/blob/main/docs/csharp/language-reference/keywords/ref.md

Programming environments for C#

C# is a Microsoft language, and requires the .NET framework (compilers and library). The good news is that the .NET framework is now cross-platform, available for Linux, Mac, and Windows. This gives you three main options for working with C# (option #1).

Option 1 : Fully Fledged IDE:
Visual Studio for Windows or Mac
Option #1.1: Visual Studio for Windows

If you are running on Windows, you can install Visual Studio 2019 Community Edition. This is a free IDE with integrated editing, compiling and running. See https://visualstudio.microsoft.com/vs/ . After installing, run "Visual Studio Installer", select "Modify", and select the workload for ".NET Desktop Development" --- make sure "C# language support" is checked. Install, and you'll have complete F# support available in one tool. Once installed, create a new C# project of type console app, against either .NET Core or .NET Framework.

Option #1.2: Visual Studio for Mac

If you are running on Mac OS X, you can install Visual Studio 2019 for Mac. This is a free IDE with integrated editing, compiling and running. See https://visualstudio.microsoft.com/vs/ , select Mac, download, and install. Once installed, create a new C# project of type console app.

Option 2 : IDE with command line hooks
Visual Studio Code (Not to be confused with the full install of Visual Studio)

Visual Studio Code is at its core a text editor, and there are a wide variety of extensions to enhance the capabilities of the environment. The good news is that Visual Studio Code is cross-platform, so it's available on Linux, Mac and Windows. When combined with the .NET framework, it can be a platform for C# programming with Intellisense, debugging, and type inference. Since the details differ by platform, here's the rough idea of what you need to do:

1. install .NET Core 3.1: https://dotnet.microsoft.com/download or https://www.microsoft.com/net/core
2. install Visual Studio Code: https://code.visualstudio.com/
3. startup Visual Studio Code program…
4. View menu, Extensions, search for "C# for Visual Studio Code (powered by OmniSharp)", install

At this point Visual Studio Code is configured to edit C# code, and provide Intellisense support and type inference information. But to create your initial C# program --- source file and makefile --- you need to open a terminal window (DOS shell on Windows, or terminal on Mac or Linux), and do the following to create a new C# program:

```
a) dotnet new console -lang C# -o HW5
```

This generates a skeleton C# "hello world" program (and makefile) in a sub-directory called "HW5". Back in Visual Studio Code, use the View menu, Explorer to view your local file system, and then use the File menu, Open Folder to navigate and open your "HW5" sub-directory. Now you should see "Program.cs" in the Explorer window pane. Click on "Program.cs" to view the skeleton code that was generated. When you are ready to compile and run this program, save any changes (Ctrl+S), switch back to the DOS shell / terminal

window, and run the following commands:

```
b) dotnet build
c) dotnet run
```

The idea is to edit with VS Code, and compile & run using the terminal window. [*NOTE: there's a way to configure "tasks" in VS Code to build and run without the need for a separate terminal window, I haven't found consistent instructions for this process across platforms, even the instructions say to run at the command line.* https://docs.microsoft.com/en-us/dotnet/fsharp/get-started/get-started-vscode]

Option 3 : **Command line compilation**
Use your own text-editor/IDE
The instructions for compiling and running C# programs on the command line match the ones used if you are using Visual Studio Code. From the terminal, execute the following once the packages have been installed.

```
a) dotnet new console -lang C# -o HW5
b) cd HW5
c) dos2unix *
```

At this point you should see two files, "Program.cs" and "program.csproj". The former is a pregenerated C# program to output "hello world", and the latter is a makefile. Once you have written your program to Program.fs, compile as follows.

```
d) dotnet build
e) dotnet run
```

You may need to run dos2unix on the Program.cs file generated by dotnet new console command, to have consistent line endings.

For reference, this is the sequence of commands we run on the docker environment in gradescope to install the C# library to that Ubuntu virtual machine

```
apt-get install dos2unix -y
wget -q https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb
dpkg -i packages-microsoft-prod.deb
add-apt-repository universe -y
apt-get install apt-transport-https -y
apt-get update -y
apt-get install aspnetcore-runtime-5.0 -y
apt-get install dotnet-sdk-5.0 -y
```

Part 1

Your exercise is to fill in the template C# code provided (HW5P1.cs) in the commented out section asking

```
// Use a switch case to call the function the user chose
// If it was none of the options, output "Invalid choice"
```

You must write switch cases for the different integer inputs provided, calling the function F#, where # is the number of the input entered by the user. Call the provided C# functions from the switch cases (F1, F2, F3..) and if a parameter is required, pass in the input as a parameter.

For example, here is what the output looks like for the following input. I've put a dash by the lines in the image on the right which were the user typing in the input.

1
5
3
10
17
4
-7
45
9
0

Just the output

```
The user selected function 1.
5
The user's input was 3
Last function
Invalid choice
4
Invalid choice
Invalid choice
Number 9
```

Inputs and outputs

```
1 -
The user selected function 1.
5 -
5
3 -
The user's input was 3
10 -
Last function
17 -
Invalid choice
4 -
4
-7 -
Invalid choice
45 -
Invalid choice
9 -
Number 9
0 -
```

Part 2

In this part of the homework, we are re-implementing Project 3, replacing the main in that F# application that only allows for selecing one task with a loop in C# that lets you continue to select tasks. The computation is still happening primarily in the provided .fs file, but will need to add the input of article id for tasks 1-3, and the output for each of the tasks to the .cs file.

The template provided for Part 2 contains a .fs file which holds the functions from Project 3, and a .cs file which includes the switch case template code from Part 1. For this part of the project, you must use the .fs file to build a Library, then create a solution which includes both the csharp console application from part 1, and the library you build for part 2. In this application, you must replace the contents of functions F1 through F10 with calls to the F# functions to compute the outputs as though the input was provided to Project 3.

Hints:

The file begins with functions commented out so you can test your solution one task at a time.

You can use var to catch the variables returned from F# functions, regardless of type.

You can use foreach(var name in collection) to iterate over collections such as FSharpLists.

Watch the 4-14 async video and sync session recordings to get instructions for how to build the solution.

On the inputs of

dataSmall.csv
1
2
2
1
3
1
4
5
6
7
8
9
10
0

The output is as listed in these screenshots

```
Enter name of the csv file containing employee data:
dataSmall.csv
Which task to perform (0 for quit):
1
Enter id of article:
2
1. Title: Everything you need to know about the coronavirus

Which task to perform (0 for quit):
2
Enter id of article:
1
2. Number of Words in The Article: 9

Which task to perform (0 for quit):
3
Enter id of article:
1
3. Month of Chosen Article: February

Which task to perform (0 for quit):
4
4. Unique Publishers:
The New York Times
National Public Radio (NPR)
Sputnik News

Which task to perform (0 for quit):
5
5. Unique Countries:
USA
Russia
```

```
Which task to perform (0 for quit):
6
6. Average News Guard Score for All Articles: 66.875

Which task to perform (0 for quit):
7
7. Number of Articles for Each Month:
        January : ************************1
       February : ************************************************2
          March : 0
          April : 0
            May : ************************1
           June : 0
           July : 0
         August : 0
      September : 0
        October : 0
       November : 0
       December : 0


Which task to perform (0 for quit):
8
8. Percentage of Articles That Are Reliable for Each Publisher:
The New York Times: 100.000 %
National Public Radio (NPR): 50.000 %
Sputnik News: 0.000 %

Which task to perform (0 for quit):
9
9. Average News Guard Score for Each Country:
USA: 85.000
Russia: 12.500

Which task to perform (0 for quit):
10
10. The Average News Guard Score for Each Political Bias Category:
        Alphabet : ****************80.000
          Center : ********************100.000
     Left-center : ***************75.000
           Right : **12.500


Which task to perform (0 for quit):
0
```

Part 3

In this part of the homework, we are re-implementing Project 4, wrapping the 10 queries that were in different .sql files with a single C# program that sends queries to MySQL Server. This C# program has a menu loop like the previous projects, allowing the user to choose which query they want to run repeatedly until they choose query 0, exiting the application.

You should use the queries from the sql files as provided in the queries folder (P4_01 to P4_10), running against the database hw5, which is populated by the provided load_datasmall.sql. Other databases will be tested in the autograder, but they will all be accessed through the database name hw5. The first query asks for a user input for which article to get the title of, none of the other queries take an additional input. For each output, output the names of the columns from the query on one line, separated by tab characters, then one line per row of the output, with each of the columns separated by tab characters.

The template provided for Part 3 contains a folder of .sql files which holds the queries from Project 4, and a .cs file which includes the menu template (while loop and switch case to call the different functions). Inside the functions, I have included comments to remind you of the steps to take to communicate with the database and produce the output. In S1 I have provided additional comments describing how to use some of the pieces (in particular the DataReader), and in S2 I have provided the query as an example of how to copy the query and commented out formatting for the output to demonstrate how to generate tab separated columns.

To build this project, you want to build a .NET Core C# console application, then add the NuGet Package for MySql.Data. This can be done in Visual Studio and Rider by right clicking on the project from the solution explorer, and then choosing to Manage NuGet packages. Alteratively, from the command line you can add the package by running the command:

```
dotnet add package MySql.Data
```

There is no additional library set up for this part of the homework, assuming that you still have MySql server running from Project 4, otherwise review the documentation for that project on how to set up MySql.

Hints:

There is going to be a lot of copy-pasted code between the 10 functions, consider creating additional helper functions to clean up your solution and standardize the method for database access.

The file begins with functions commented out so you can test your solution one task at a time.

An example of executing a query can also be found in the official documentation

https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-sql-command.html

Watch the 4-16 async video and latter halves of the sync session recordings to get instructions for how to build the solution.

You can rename your source code file from Program.cs to HW5P3.cs in the IDE to avoid conflicts with the autograder.

Make sure you've reverted back to the original connection string from the template before submitting the code to Gradescope if you've changed the username, password, or database to test your code.

On the inputs of

```
1
2
2
3
4
```

5
6
7
8
9
10
0

The output is as shown in the following screenshots

```
Testing Connection to MySQL Server...
Established Connection to MySQL Server.
Which task to perform (0 for quit):
1
Enter a news id:
58
title
The Coronavirus: What Scientists Have Learned So Far

Which task to perform (0 for quit):
2
news_id length
2       235

Which task to perform (0 for quit):
3
title   Month
Bagels  March
Everything you need to know about the coronavirus       February
The Coronavirus: What Scientists Have Learned So Far    January
Chinese Health Officials: More Die From Newly Identified Coronavirus    February

Which task to perform (0 for quit):
4
publisher
National Public Radio (NPR)
The New York Times
Whispers of Washington

Which task to perform (0 for quit):
5
country articleCount
USA     3
UK      1
India   0
```

```
Which task to perform (0 for quit):
6
Average Score
73.750

Which task to perform (0 for quit):
7
month   numArticles     overall percentage
January 1       4       25.000
February        2       4       50.000
March   1       4       25.000

Which task to perform (0 for quit):
8
publisher       percentage
The New York Times      100.000
Whispers of Washington  100.000
National Public Radio (NPR)     50.000

Which task to perform (0 for quit):
9
country avg_news_score
USA     85.000
UK      40.000

Which task to perform (0 for quit):
10
author  political_bias  numArticles
Emily Feng      Center  1
Knvul Sheikh    Alphabet        1
Knvul Sheikh    Left-center     1
Nicole Wetsman  Left-center     2
Roni Caryn Rabin        Alphabet        1

Which task to perform (0 for quit):
0
```

Electronic Submission

When you're ready, submit your source code file "HW5P1.fs" on Gradescope under "Homework 5 Part 1".

For part two, submit only one source code file, "HW5P2.cs" on Gradescope under "Homework 5 Part 2". Your solution should work without any modifications to the library.

For part three, submit the one source code file "HW5P3.cs" on Gradescope under "Homework 5 Part 3". Ensure that the connection string in your application has the user name root, the password password, and connects to the database hw5 (capitalization matters).

Policy

Late work is not accepted for this assignment after the final deadline. All submissions should occur before noon. All work is to be done individually — group work is not allowed. You are free to discuss the language of C# and the constructs and examples used in lectures and sync sessions on Piazza. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml .