

Project #04: Divvy Daily Ride Analysis

Complete By: Friday March 9th @ 11:59pm
Assignment: F# program to perform Divvy daily ride analysis
Policy: Individual work only, late work **is** accepted
Submission: via Blackboard (“Projects”)

Divvy Bike Sharing

You’ve probably seen the blue bikes that are part of Divvy bike sharing system here in Chicago. The Divvy system is a commuting service where users pay a yearly fee (\$75?) to use the bikes for free as long as rides are < 30 minutes in duration. Longer rides incur a surcharge. It’s a very popular system, with stations throughout the city and some suburbs.



Interestingly, the Divvy folks also make their ride data available as part of Chicago’s open data [portal](https://www.divvybikes.com/data); Divvy’s data is available @ <https://www.divvybikes.com/data>.

The goal in this assignment is to input ride data for one day, and perform some analysis of the data: # of rides, average ride duration, etc. Here’s a screenshot of the program analyzing the **day1.txt** data file:

- 4711 rides
- 73.42% of the riders identify as male
- 26.58% of the riders identify as female
- Average ride is 10.71 minutes long
- Average age of male-identified riders is 37
- Average age of female-identified rides is 35.88

The program starts by inputting the filename, and then proceeds to input and analyze the data. A skeleton F# program is provided with the code to open the file and input the data. Your job is to add the analysis.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd...'. The command 'day1.txt' is entered and highlighted with a red box and a red arrow. The output of the program is displayed in green text: '# of rides: 4711', '% of male riders: 73.42390151', '% of female riders: 26.57609849', 'Avg duration: 10.70763461 mins', 'Avg age of male riders: 37.00491472', 'Avg age of female riders: 35.88498403', and 'Press any key to continue . . .'.

```
C:\WINDOWS\system32\cmd...  
day1.txt  
# of rides: 4711  
% of male riders: 73.42390151  
% of female riders: 26.57609849  
Avg duration: 10.70763461 mins  
Avg age of male riders: 37.00491472  
Avg age of female riders: 35.88498403  
Press any key to continue . . .
```

File format

The input files consist of at most 5000 “rides”, representing one day of ride data. Here’s a snapshot of the provided **day1.txt** input file:

```
3808,311,1,1984
1756,309,1,1991
253,524,1,1973
.
.
.
27,1063,2,1988
```

Each ride consists of 4 values: bike id, duration, rider’s gender, and rider’s birthyear.

- Bike id: integer
- Trip duration: integer, in seconds
- Gender: integer, gender that rider identifies with, 1=>male and 2=>female
- Birth year: integer, year that rider was born (e.g. 1991)

Along with the skeleton F# program we are providing 4 input files: **day0.txt** is a small file with only 24 entries, for easier testing. Then 3 real files: **day1.txt**, **day2.txt**, and **day3.txt**.

Getting Started

To help you get started, we are providing an F# console program in the form of a Visual Studio solution. To start, browse to the course web [site](#), open Projects, and then open [project-04-files](#). Download the file “DivvyDailyRides.zip”:

<https://www.dropbox.com/s/3v24vvj9di3y1g8/DivvyDailyRides.zip?dl=0>

After you download, open the .zip, extract the folder “DivvyDailyRides”, and save to your Desktop. Close and immediately delete the .zip so you don’t work with it accidentally.

Open the **DivvyDailyRides** folder, and double-click on the Visual Studio Solution (.sln) file to open the program in VS. Via the Solution Explorer window pane, open the F# source code file, and you’ll see the following:

```
//
// F# program to analyze Divvy daily ride data.
//
// <<your name>>
// U. of Illinois, Chicago
// CS 341, Spring 2018
// Project #04
//
```

```
#light

let ParseLine (line:string) =
    let tokens = line.Split(',')
    let ints = Array.map System.Int32.Parse tokens
    Array.toList ints

let rec ParseInput lines =
    let rides = Seq.map ParseLine lines
    Seq.toList rides

[<EntryPoint>]
let main argv =
    //
    // input file name, then input divvy ride data and build
    // a list of lists --- [ [1308;321;2;1991]; ... ]
    //
    let filename = System.Console.ReadLine()
    let contents = System.IO.File.ReadLines(filename)
    let ridedata = ParseInput contents

    //printfn "%A" ridedata
    let N = List.length ridedata
    printfn "# of rides: %A" N
    //
    // TODO:
    //
    0
```



Notice that the **main** function inputs the filename, then opens the file and inputs the data. Each line of the file --- one ride --- is turned into a list of 4 items. So the resulting data structure is a list of lists. If you uncomment the line denoted above, you can see the format of the input after parsing. Here's what you'll see working with the small input file **day0.txt**:

```
C:\WINDOWS\system32\cmd.exe
day0.txt
[[3808; 311; 1; 1984]; [1756; 309; 1; 1991]; [253; 524; 1; 1973];
[253; 423; 1; 1973]; [881; 477; 1; 1990]; [519; 535; 1; 1972];
[1756; 321; 1; 1993]; [1056; 1508; 1; 1987]; [253; 477; 1; 1973];
[391; 1234; 1; 1982]; [4491; 480; 1; 1985]; [1713; 336; 1; 1993];
[2829; 195; 1; 1995]; [3930; 972; 2; 1990]; [1967; 579; 1; 1992];
[2381; 823; 1; 1985]; [2366; 853; 2; 1985]; [4721; 943; 2; 1988];
[95; 1082; 2; 1992]; [1575; 1018; 1; 1986]; [2625; 955; 2; 1985];
[378; 1885; 2; 1986]; [1324; 884; 1; 1984]; [27; 1063; 2; 1988]]
# of rides: 24
Press any key to continue . . .
```

Requirements

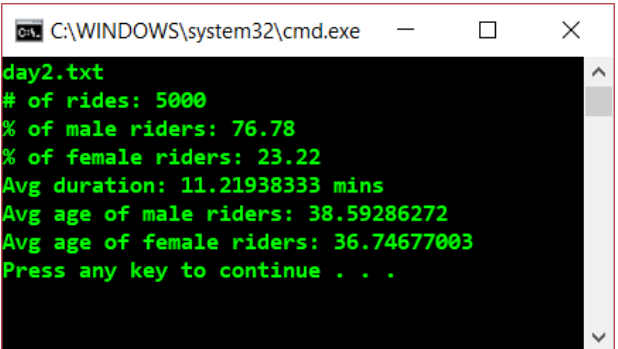
No imperative programming. In particular, that means no mutable variables, and no loops. Use recursion and higher-order approaches only.

A few F# hints / gotchas...

Sometimes when I open an F# program in Visual Studio, I get a very strange error: all the code is underlined and VS complains that I'm "missing a reference to mscorlib". I close the solution and reopen it, and the error goes away.

To the right is the correct output for processing **day2.txt**. Notice that 2 of the outputs start with "%". Since "%" has special meaning to printfn, you have to escape % to print --- so put 2 in a row, i.e. "%%". Also, notice most of the results are real numbers, while the input is integer-based. I suggest you work with integers, and when it comes time to compute a real number result, convert each integer to a double using the **double** function. Example: suppose you have an integer sum and an integer N. To compute the average as a real number, do

```
let avg = (double sum) / (double N)
```



```
C:\WINDOWS\system32\cmd.exe
day2.txt
# of rides: 5000
% of male riders: 76.78
% of female riders: 23.22
Avg duration: 11.21938333 mins
Avg age of male riders: 38.59286272
Avg age of female riders: 36.74677003
Press any key to continue . . .
```

Submission

Before you submit, make sure your name appears in the header comment at the top of the F# source file. Then close Visual Studio and locate the top-level project folder **DivvyDailyRides**. Right-click on the top-level project, Send to, and select "Compressed (zipped) folder". The result will be a single archive (.zip) file that you can then submit to Blackboard. Submit under "Projects", "P04 Divvy Daily Ride Analysis".

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. via Piazza), 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 described here:

<http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf> .

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. 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 <http://www.uic.edu/depts/dos/studentconductprocess.shtml> .