Homework #5

Complete By: Monday, October 5th @ 11:59pm
Assignment: completion of following exercise
Policy: Individual work only, late work *is* accepted
Submission: electronic submission via Blackboard

Overview

We are going to analyze a file containing rainfall data. The data was collected @ Chicago's Midway airport, and looks as follows:

2002	**0.01**	**1.77**	**0.78**	**3.02**	**5.17**	**5.84**	**4.29**	**1.12**	**4.06**	**1.9**	**2.07**	**1.47**	**1.53**
2001	**0.02**	**1.24**	**3.23**	**1.61**	**3.33**	**4.55**	**2.4**	**3.5**	**7.32**	**3.9**	**6.04**	**1.69**	**1.13**
2000	**0.01**	**1.56**	**1.4**	**1.17**	**4.74**	**6.48**	**4.81**	**4.07**	**3.19**	**6.2**	**1.95**	**2.65**	**2.9**

The file contains N > 0 lines of data. Each line contains 14 values separated by tabs: the year, error factor, and the amount of rainfall measured for each month (January, February, …, December). You may assume the years will be in descending order, but the starting year may change, as well as the total # of years; ignore the error factor.

The name of the file is always "rainfall-midway.txt", and a sample file can be downloaded via the "Homeworks" link from the course web page: http://www.joehummel.net/cs341.html. Your task is to read this file, and output the following statistics:

- Average rainfall for each year
- Average rainfall for each month (across all years in the dataset)
- Max rainfall in dataset, along with month and year; if there's a tie, any max value is fine
- Min rainfall in dataset, along with month and year; if there's a tie, any min value is fine

For the dataset above, here is the correct output, and the required output format:

```
** Rainfall Analysis Program **

2002: 2.751666667
2001: 3.328333333
2000: 3.426666667
```

```
Jan: 1.523333333
Feb: 1.803333333
Mar: 1.933333333
Apr: 4.413333333
May: 5.623333333
Jun: 3.833333333
Jul: 2.896666667
Aug: 4.856666667
Sep: 4.0
Oct: 3.353333333
Nov: 1.936666667
Dec: 1.853333333

Max rainfall:  7.32, 8/2001
Min rainfall:  0.78, 2/2002

** Done **
```

Assignment Details

Create an F# console app using Visual Studio 2013/2015.

- You are free to use the functions available to you in F#.
- Do not use mutable variables — doing so will result in an immediate 0. Write functional code.
- For testing, you can download a "rainfall-midway.txt" file from the course web page. Place this file into your Visual Studio solution folder as follows. Suppose your project is named "Rainfall". Create your project, build it, and then minimize Visual Studio. Now drill-down into the folder Rainfall\Rainfall\bin\Debug, and place a copy of the input file in that folder:

You know you are in the right place if your input file is in the same folder as your .exe file. This

assumes you are working in "Debug" mode; if you switch to "Release" mode, you'll need to put another copy in Rainfall\Rainfall\bin\Release.

Starter code

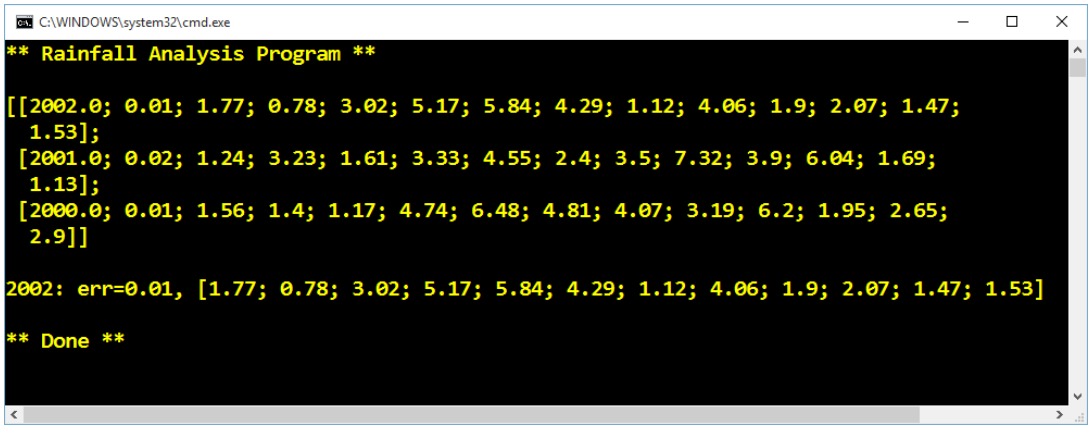
Here's some code to help you get started. In particular, you'll see functions to read the input file into a list of strings, parse that data into a list of lists of doubles, and then output the first year of data in a more readable way. You'll also want to note in particular this output function:

```
let PrintOneYearOfData (values: double list) =
  printfn "%A: err=%A, %A" (int values.Head) values.Tail.Head values.Tail.Tail
```

One year of data is represented as a list of doubles, e.g. `[2002.0; 0.01; 1.78; 0.77; …]`. The function above shows you how to (a) provide type information such as (values: **double list**) in cases where F# cannot infer the type correctly (the error message you'll get in these case is "*Lookup on object of indeterminate type… A type annotation may be needed…*"), and (b) how to convert the year from double to int for better printing. This code is available for download via the "Homeworks" link on the course web page, see "hw5-program.txt":

```
#light

//
// ReadFile: reads a file line by line, returning a list
// of strings, one per line.
//
let ReadFile filename =
  [ for line in System.IO.File.ReadLines(filename) -> line ]

//
// ParseLine: given a line from the rainfall file, parses this
// string into list of double values starting with the year, error
// rate, and then the 12 rainfall values (one per month).
//
let ParseLine (line:string) =
  let strings = line.Split('\t')
  let strlist = Array.toList(strings)
  let values = List.map (fun x -> System.Double.Parse(x,
System.Globalization.NumberFormatInfo.InvariantInfo)) strlist
  values

//
// Given a year of data as a list of double values, outputs the year,
// error rate, and then the 12 rainfall values for each month.
//
let PrintOneYearOfData (values: double list) =
  printfn "%A: err=%A, %A" (int values.Head) values.Tail.Head values.Tail.Tail
```

```
//
// Main:
//
[<EntryPoint>]
let main argv =
  printfn "** Rainfall Analysis Program **"
  printfn ""

  //
  // read entire file as list of strings:
  //
  let file = ReadFile "rainfall-midway.txt"

  //
  // Parse the data into a list of lists, where
  // each sub-list is [year err value value ...]:
  //
  let rainfall = List.map ParseLine file

  // debugging:
  printfn "%A" rainfall
  printfn ""

  //
  // Print first year of data:
  //
  PrintOneYearOfData rainfall.Head

  // done
  printfn ""
  printfn "** Done **"
  printfn ""
  printfn ""
  0 // return 0 => success
```



Electronic Submission

Using Blackboard (http://uic.blackboard.com/), submit your "Program.fs" source code file under the assignment "HW5". Attach the file, do not copy-paste. If you have multiple files, place all files needed to build your program into a folder, create an archive file (e.g. a .zip), and submit that archive file instead. The top of your main source file should contain a header comment like the following:

```
//
// F# program to analyze rainfall data.
//
// <<YOUR NAME HERE>>
// U. of Illinois, Chicago
// CS341, Fall 2015
// Homework 5
//
```

Your program should be readable with proper indentation and naming conventions; commenting is expected where appropriate. You may submit as many times as you want before the due date, but we grade the last version submitted. This implies that if you submit a version before the due date and then another after the due date, we will grade the version submitted after the due date — we will *not* grade both and then give you the better grade. We grade the last one submitted. In general, do not submit after the due date unless you had a non-working program before the due date.

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. your "iClicker teammates" or 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. 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 http://www.uic.edu/depts/dos/studentconductprocess.shtml .