CSc 372, Fall 2006
Assignment 5, Part 1
Due: Thursday, November 2 at 23:59:59

A Note About Grading

We're still seeing a lot of cosmetic errors in submitted solutions. Mistakes like misnamed files, extra whitespace, and text in the wrong case are easily detected with the supplied tester. **Beginning with this assignment, if you submit a solution with an error that could have been detected with the tester, it is unlikely that the assigned score, perhaps a zero, will be changed**. This may seem harsh but in industry a single bit can be the difference between a successful product release and a big mess, or worse. There's no better time than now to get in the habit of using automated testing whenever possible. In short, use the tester!

Two-Part Assignment

This assignment is being issued in two parts, both of which will be due on November 2. The second part should be available on the web no later than Sunday, October 22. There will be a total of 97 regular points worth of problems and a few extra credit points, too.

Problem 1. (45 points) `mtimes.rb`

For this problem you are to create a Ruby program that reads a text file with data on theaters, movies, and showtimes and then creates an HTML document that shows when and where movies of interest are showing. Here is an example of the data:

```
% cat mtimes.0
Century Park
    Everyone's Hero [G] [NP] (11:35a), 1:45, 4:05, 6:15, 8:05
    Gridiron Gang [PG-13] [NP] (12:25), 1:30, 3:20, 4:35

Gateway
    Snakes on a Plane [R] 11:40a, 12:45, 2:05, 5:30, 9:20
    Monster House [PG] 11:45a, 12:50, 2:00, 6:30, 7:35, 8:45

Park Place
    The Guardian [PG-13] 3:15, 7:00
    Gridiron Gang [PG-13] [NP] (12:00) 1:30, 3:00, 4:35, 5:55
    The Covenant [PG-13] [NP] (12:25), 2:55, 4:20, 5:30, 8:00
```

We might run `mtimes` like this:

```
% ruby mtimes.rb mtimes.0 the gang snakes
%
```

This invocation indicates that the data file `mtimes.0` should be used and that the user wants to see information on movies with titles that contain any of these strings: `the`, `gang`, and `snakes`.

As shown by the prompt immediately following the invocation above, `mtimes.rb` produces no output on standard output in the normal case. Instead it creates an HTML document that is always named `mtimes.htm`.

The invocation

```
ruby mtimes.rb mtimes.0 the gang snakes
```

creates a copy of `mtimes.htm` that in a browser looks like this:

Gridiron Gang		
	Century Park	Park Place
12:00		•
12:25	•	
1:30	•	•
3:00		•
3:20	•	
4:35	•	•
5:55		•

Snakes on a Plane	
	Gateway
11:40	•
12:45	•
2:05	•
5:30	•
9:20	•

The Covenant	
	Park Place
12:25	•
2:55	•
4:20	•
5:30	•
8:00	•

The Guardian	
	Park Place
3:15	•
7:00	•

A table of showtime and theater information is displayed for each matching movie, ordering the tables alphabetically by movie title. A movie is never displayed more than once, no matter how many of the specified strings match the title. In a table for a particular movie, theaters are in alphabetical order.

A movie is matched by a string if the string appears anywhere in the title. Case is ignored. For example,

```
ruby mtimes.rb mtimes.0 hE
```

would produce tables for *Everyone's Hero*, *The Covenant*, and *The Guardian*, in that order.

A very wide document will be produced if a string matches a lot of titles. For example, "`e`" matches quite a few. Don't worry about that; assume that wide document is exactly what the user wants to see.

If none of the specified strings match a movie or if no strings are specified, "`No movies matched`" is printed:

```
% ruby mtimes.rb mtimes.1 nosuchmovie
No movies matched
% ruby mtimes.rb mtimes.1
No movies matched
%
```

There will be only two data files used for grading purposes: `mtimes.0` and `mtimes.1`, both in `/home/cs372/fall06/a5`. Both are already present and neither file will change unless a must-fix error is discovered. **Your program should be able to handle all the data in those two files but there's no need to contemplate "what-if"s with other data files.** Assume that a data file is always specified and that it exists.

The expression `htmfile = File.new("mtimes.htm", "w")` will open `mtimes.htm`, creating it if it doesn't exist. Assume that the open is successful. You can write to the file with `htmfile.puts/print/printf`, etc.

VERY IMPORTANT: Solutions will be NOT be graded by viewing them with a browser. Instead we'll `diff` the resulting `mtimes.htm` files against those produced by my solution. This implies that you've got to produce exactly the same HTML as my solution. That's not hard but it implies that you must check your output with `tester` (already in place) to be sure your output exactly matches mine.

Note that the HTML files that `tester` diffs against are in

```
/home/cs372/fall06/a5/master/tester.out/mtimes.out.*
```

For your convenience, the command

```
/home/cs372/fall06/a5/gethtms
```

copies the current batch of `mtimes.out.*` files into `mtimes1.htm`, `mtimes2.htm`, etc., in the current directory.

You may be inclined to throw some object-oriented design at this problem, creating classes like `Movie`, `Theater`, `Showing`, etc. I'd say that's overkill. This problem is intended to be an exercise in working with hashes, arrays, and regular expressions. Also, we haven't and probably won't cover the issues that one may encounter when doing things like indexing hashes with instances of classes you define yourself.

If you don't know HTML, don't panic. Whether you know or it not you'll probably want to start by looking at the HTML that my solution produces. It's very simple but you have any trouble understanding the structure of it, please let us know.

The point value on this problem is somewhat inflated due to the hassle of dealing with HTML and having to exactly match my output. Were it not for that this would perhaps be a 35 point problem. In terms of ballpark size, I expect that good solutions will require 100-200 lines of code.

Pair Programming

Subject to the rules below you may develop `mtimes.rb` using Pair Programming. `mtimes.rb` is sized for one person but I'm allowing this to see how working in pairs changes the development experience when working in Ruby.

Here are the rules:

- 100% of on-computer time on a problem must be done by two students working at one computer.
- Both students `turnin` identical copies of the solution.
- Keyboard time should ideally be split 50/50 between partners but up to 60/40 is acceptable. You must keep a log of keyboard time with a resolution of 15 minutes. Include that log as comments in your solution.
- If you start working as a pair and elect to not finish as a pair, all code developed as a pair must be discarded. Similarly, if you start working on your own and decide that you'd like to pair, neither you or your partner can bring any code into the pair—you must start from scratch as a pair.
- If you want to pair and can't find a partner, put an ad on the class mailing list.

Pair Programming may be allowed on some of the part 2 problems, too.

Make Your Own Assignment

If you've got an idea for something you'd like to write in Ruby you can propose that as a replacement for some or all of the problems on this assignment. To pursue this option send mail to `cs372-staff` with a brief sketch of your idea. We'll negotiate on points and details. Pairing is permissible.

It is important to note that it's virtually impossible to be completely equitable when settling on how many points a problem is worth. Be sure to recognize that you might end up working harder for your points than those doing the regular problems or another student doing a problem of his/her own design. Similarly, those doing the regular problems should recognize that good negotiators may end up with less work.

Miscellaneous

An underknown fact is that one way to open an HTML file in a browser is to use `Start > Run...` and specify the path, perhaps `h:\372\a5\mtimes.htm`. Or, enter the pathname in a command prompt window.

The second part of the assignment will contain information on deliverables and such.