Baby Names Project

The Social Security administration has this neat data by year of what names are most popular for babies born that year in the USA (see social security baby names).

The files baby1990.html baby1992.html ... contain raw html, similar to what you get visiting the above social security site. Take a look at the html and think about how you might scrape the data out of it.

Part A

Implement the ***extractNames(filename)*** function which takes the filename of a **baby1990.html** file and returns the data from the file as a single list(array) -- the year string at the start of the list followed by the name-rank strings in alphabetical order. **['2006', 'Aaliyah 91', 'Abagail 895', 'Aaron 57', ...]**. In your main method, call your ***extractNames()*** function and prints what it returns. If you get stuck working out the regular expressions for the year and each name, solution regular expression patterns are shown at the end of this document. Note that for parsing webpages in general, regular expressions don't do a good job, but these webpages have a simple and consistent format.

Rather than treat the boy and girl names separately, we'll just lump them all together. In some years, a name appears more than once in the html, but we'll just use one number per name. (*Optional: make the algorithm smart about this case and choose whichever number is smaller.)*

Build the program as a series of small milestones, getting each step to run/print something before trying the next step. This is the pattern used by experienced programmers -- build a series of incremental milestones, each with some output to check, rather than building the whole program in one huge step.

Printing the data you have at the end of one milestone helps you think about how to re-structure that data for the next milestone. Here are some suggested milestones:

- Extract all the text from the file and print it
- Find and extract the year and print it
- Extract the names and rank numbers and print them

- Get the names data into a data structure of your choice (HashMap) and print it
- Build the [year, 'name rank', ...] list and print it
- Fix main() to use the ExtractNames list

Earlier we have had functions just print to standard out. It's more re-usable to have the function return the extracted data, so then the caller has the choice to print it or do something else with it. (You can still print directly from inside your functions for your little experiments during development.)

Have ***main*** call ***extractNames()*** for each command line arg and print a text summary.

The summary text should look like this for each file:

```
2006
Aaliyah 91
Aaron 57
Abagail 895
Abbey 695
Abbie 650
...
```

Part B

Suppose instead of printing the text to standard out, we want to write files containing the text. If the flag or command-line argument “summaryfile” is present, do the following: for each input file 'foo.html', instead of printing to standard output, write a new file 'foo.html.summary' that contains the summary text for that file.

Once the “summaryfile” feature is working, run the program on all the files.

With the data organized into summary files, you can see patterns over time with shell commands(In your Terminal, run **`man grep`** for more information), like this:

```
$ grep 'Trinity ' *.summary
$ grep 'Nick ' *.summary
$ grep 'Miguel ' *.summary
$ grep 'Emily ' *.summary
```

Regular expression hints -- year: r'Popularity\sin\s(\d\d\d\d)' names: r'<td>(\d+)</td><td>(\w+)</td>\<td>(\w+)</td>'