RMIT UNIVERSITY

Computer Science and Information Technology

COSC1292/1293 Scripting Language Programming

Semester 1 2016 – Assignment 1 – Perl

Introduction

This is **a group assignment** worth 20% towards your final grade. Each group contains no more than two members.

You are required to write at least one script and a report to demonstrate your ability of Perl programming including text analysis, file handling and building modules.

You should submit your assignment no later than 4pm 18th April 2016.

Academic Integrity [more]

The submitted assignment must be your own work. No marks will be awarded for any parts, which are not created by the group.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools may be used for all submissions. Penalties will be applied in cases of plagiarism.

General Requirements

- Your coding style should conform to general Perl coding conventions. Your scripts should be clean, neat, and well-formatted (e.g. indention). Identifiers should also be properly named.
- Your scripts should be well-documented by using POD so user can easily understand your scripts and their usage by exporting POD as HTML, plain text or LaTeX.
- You should always properly validate user inputs (including command-line arguments) and display user-friendly error messages where applicable.
- Any standard Perl modules shipped with Perl 5.18 may be used. However, you should avoid use extra modules which are not installed on CS servers.

Event Extractor

Extracting time information from plain text such as emails is a very useful functionality, but still remains as a challenge. You can see examples in Gmail. Gmail underlines date and time in attempt to create calendar events automatically. As you can see, the result is often mixed. There are many scenarios that Gmail cannot handle well. Your task here is to write a script to achieve similar and even better performance than Gmail event extractor.

Two JSON files are provided with this assignment. JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format, which is much easier to read and write comparing with XML.

The first file "`emails.json`" contains several emails. Below is an example email which shows send date/time, subject, time zone and the body:

```
{   "sent": "2016-02-01T19:03:02.00Z",
    "subject": "Upcoming events: Orientation 2016",
    "timeZone": "Australia/Melbourne",
    "content": "We're pleased to …...."
},
```

Your script is expected to accept any JSON files in that format. It should be able to extract information related to time and date. You can assume the email file is always in correct format. Hence there is no need to validate data. However usual file handling practice should not be omitted.

Your script will be tested on unseen email files.

Your script is expected to produce an output file, which clearly shows time info of events in JSON. The second file "`events.json`" is the example of such output file that your script should be generating.

Note "datetime" and "date" are different. Attribute "date" is to represent all-day event. One event cannot have both date & datetime attributes.

Each event has a start time and an end time. The default duration of a "datetime" event is 1 hour (when the end time is unspecified).

You are required to write a brief report to explain your implementation and discuss your ideas. Marks will be given to good discussions even if the coding is relatively primitive.

Evaluation

Members from the same group will receive same score by default. Members may specify their contributions in the submission, so marks will be given accordingly but capped at 20. For example two members of an 18-mark submission with 40-60 contributions will receive 14.4 (18x2x40%) and 20 marks respectively.

Individual submission is acceptable but does not attract extra marks.

In total 100 marks are allocated to the following tasks.

15 marks, io: your script can read any email JSON files assuming data format is always correct.

Your script can produce an output file "events.json" using the given JSON format. You MUST specify UTC time in output file. https://www.w3.org/TR/NOTE-datetime

Command line arguments can be handled properly.

10 marks, basic: your script can produce correct output for at least two events using proper regular expression.

10 marks, style: for good coding style such as meaningful variable names, comments, use of POD.

10 marks module: the main event extraction functionalities are modularized and packaged as one PERL module for example "eventExtractor.pm".

10 marks, all day events: your event extractor can differentiate normal events and all-day events, and produce correct JSON output for all day events.

10 marks, time zones: your script can handle different time zone.

If the timezone is not specified, use the local timezone.

Timezone API: https://timezonedb.com/api, you can use key `EHUCC69JBOJT` or register your own account. Students are free to use this API.

Daylight saving may need to be considered.

10 marks, relative time: your event extractor can handle relative time as well as absolute time, for example, "tomorrow 5pm", "next Tuesday" and "Monday 9am".

10 marks, multi format: your event extractor can cope with multiple forms of time information for example, " 12-Dec-2015", " 2015 01 20", "Apr. 19th 98", "6pm", "23:00", "next Wed. 12pm".

You are not expected to handle time related information like "this afternoon", "tomorrow night", "every Tuesday", "fortnightly" and so on.

15 marks, report: This is a mini research document, which explains your implementation, presents your findings, thoughts and ideas, and discusses how your code can be improved and how the above tasks can be handled more effectively. No more than 5 pages.

NOTE, we are not expecting a perfect event extractor, which may be no simpler than AlphaGo.

A competition will be held using unpublished email data. The winning team will receive rewards and bonus marks. Details will be announced later.

Submission

Submission should be made available on Blackboard. The deadline is **4pm Monday 18th April 2016**.

You can submit your assignment as many times as you want before the due date. The last submission will overwrite the previous.

Late submissions will incur a penalty of 10% per day. Submissions made 5 days after the due date will receive no marks.

Extensions will only be permitted in exceptional circumstances. Workload and/or heavy load of assignments will not be accepted as exceptional circumstances for an extension.