Functional Programming
Project: Doodle (part 2)

Assistant: Noah Van Es
noah.van.es@vub.be

The final grade of the course is determined by an oral exam and a project (each counts for 50% of the total grade for the course). The project consists out of two parts. A non-empty solution for both parts has to be submitted in order to get a grade. This document describes the second part of the project.

Project description

The goal of this programming assignment is to extend the doodle system of the first assignment to better handle the scheduling of exams "by appointment". At first, teachers are expected to create a doodle for each one of their exams. Then, students can subscribe to these exams and vote for their preferred exam timeslots. The system can then output an *exam schedule* which is a mapping from exams to timeslots such that:

- Timeslots belong to the associated doodle that was proposed by the teacher at first.
- Teachers do not have to take two exams at the same time.
- Students do not have to participate in two exams at the same time.

Your system should be able to find the exam schedule (if any can be found) that scores the highest amongst all the student's votes.

Practical information

Submission The deadline for this assignment is **June 19, 2022 (23:59)**. You submit all your code in single ZIP-file through the Canvas platform.

Grading Note that your project will not only be graded on functionality, but also on the efficiency and quality of your implementation (programming style, performance considerations, ...).

Individual work In case you have questions or need clarification to complete this project, feel free to contact the assistent by mail (noah.van.es@vub.be) or make an appointment to schedule a meeting.
The project has to be executed on a strictly individual basis! Automated tools are used to scan for plagiarism between projects; any suspicion of code exchange shall immediately be reported to the faculty's dean, and both parties shall be sanctioned in accordance with the examination rules regarding plagiarism.

Project assignment

For this assignment, you will have to create a TCP server that implements the protocol detailed in Table 1. Your server should handle TCP requests concurrently using the module `Network.TCP` and the function `Control.Concurrent.forkIO`. Each TCP connection is expected to handle exactly one client request and one server response, after which it should be closed. Data race conditions should be handled using software transactional memory from the module `Control.Concurrent.STM`, as seen in the lectures. To manage access rights, requests should always embed a user login. If the user identifier does not exists for the requested role or if the password does not match, the server should return the message `wrong-login`.

Concretely, the requests that your server should handle are listed below.

`add-student` and `add-teacher` : enables the administrator to add a student or a teacher to the system. If the requested user identifier is already taken, the server should return the message `id-taken`. If the requested user identifier is available, it should return a randomly generated password of at least 4 characters. The administrator login should be passed as command-line arguments during startup.

```
=> add-teacher admin@1234 walter
<= ok f1Qs
=> add-student admin@1234 jesse
<= ok 2bYo
```

`change-password` : enables users to change their password.

```
=> change-password walter@f1Qs foobar
<= ok
```

`set-doodle` : enables teachers to create new exam doodles. Unlike in the previous assignment, an exam doodle may contain overlapping timeslots. If the teacher already defined a doodle for the exam, the new doodle should overwrite the old one. If another teacher already claimed the exam identifier, the server should return the message `id-taken`.

```
=> set-doodle walter@foobar Cooking [
  2022-01-04T14:00+01:00 / 2022-01-04T16:00+01:00,
  2022-01-04T13:00+01:00 / 2022-01-04T15:00+01:00
]
<= ok
```

`get-doodle` : enables users to get the doodle associated to an exam. If the exam does not exist, the server should return the message `no-such-id`. The timeslots should be sorted by starting time.

```
=> get-doodle Cooking
<= ok [
  2022-01-04T13:00+01:00 / 2022-01-04T15:00+01:00,
  2022-01-04T14:00+01:00 / 2022-01-04T16:00+01:00
]
```

`subscribe` : enables students to subscribe to an exam.

```
=> subscribe jesse@2bYo Cooking
<= ok
```

`prefer` : enables subscribed students to vote for their preferred exam timeslot. If the student is not subscribed to the exam, the server should return the message `not-subscribed`. For each exam they are subscribed to, the student may only vote for one preferred timeslot. A second vote for the same exam simply overwrites the first vote.

```
=> prefer jesse@2bYo Cooking 2022-01-04T13:00+01:00/2022-01-04T15:00+01:00
<= ok
```

`exam-schedule` : enable a user to get the current best exam schedule. If there is no exam schedule possible, the server should return the message `no-possible-exam-schedule`.

```
=> exam-schedule
<= {Cooking : 2022-01-04T13:00+01:00/2022-01-04T15:00+01:00}
```

Grammar Rule			Comments
REQUEST	→	add-teacher LOGIN TOKEN	The administrator LOGIN adds the teacher TOKEN to the system.
		add-student LOGIN TOKEN	The administrator LOGIN adds the student TOKEN to the system.
		change-password LOGIN TOKEN	Change the password of LOGIN to TOKEN.
		get-doodle LOGIN TOKEN	User LOGIN gets the doodle of the course TOKEN.
		set-doodle LOGIN TOKEN DOODLE	Teacher LOGIN sets the doodle of the course TOKEN to DOODLE.
		subscribe LOGIN TOKEN	Student LOGIN subscribes to the course TOKEN.
		prefer LOGIN TOKEN SLOT	Student LOGIN prefers to pass the exam of course TOKEN at timeslot SLOT
		exam-schedule LOGIN	User LOGIN gets the current optimal exam schedule.
RESPONSE	→	wrong-login	Applicable for all requests.
		ok TOKEN	Applicable for: add-student and add-teacher.
		ok DOODLE	Applicable for: get-doodle.
		ok SCHEDULE	Applicable for: best-schedule.
		ok	Applicable for: change-password, subscribe and prefer.
		id-taken	Applicable for: add-teacher, add-student and set-doodle.
		no-such-id	Applicable for: get-doodle, subscribe and prefer.
		no-such-slot	Applicable for: prefer.
		not-subscribed	Applicable for: prefer.
		no-possible-exam-schedule	Applicable for: exam-schedule.
LOGIN	→	TOKEN @ TOKEN	A login is an identifier paired with a password.
DOODLE	→	[SLOTS]	A doodle is a list of timeslots.
SLOTS	→	, SLOT SLOTS	
		ϵ	
SLOT	→	TIME / TIME	A timeslot is defined by a start time and an end time.
SCHEDULE	→	{ EXAMS }	
EXAMS	→	TOKEN : SLOT , EXAMS	A schedule is a mapping from exam identifiers to timeslots.
		ϵ	

Table 1: Client - server exchange protocol. Any white space in the grammar rules can be replaced by a combination of white spaces, tabs and newline characters. TOKEN is any group of letters uninterrupted by white spaces, tabs and newline characters. TIME follows to ISO 8601 format – e.g: 2022-11-30T10:28+01:00.