B4- Functional Programming

B-PAV-360

Train Manager

A ticket to ride

0.5

Train Manager

A ticket to ride

binary name:	trainmanager
repository name:	OCAML_2016_trainmanager
repository rights:	ramassage-tek
language:	OCaml
group size:	2
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The Train Manager project consists of creating a train management system that allows you to organize trips between several stations, different types of trains, manage various conflicts (dates, times, platforms etc.), train station occupancy depending on the platform number,...

You won't have to take a real railroad into consideration, but we hope that you will employ the complexity of such a system.

Here are the different constraints that you will have to take into account:

1. there are three types of trains: TGV, Eurostar, Thalys,
2. each type of train features an average speed, which will be used to calculate the times,
3. each train has a list of stations they can stop in (for example, the Eurostar only goes to Brussels, Lille, London and Paris),
4. the train stations are not all connected by a direct line. The train station network, with the distances between stations, is given at the end of this document,
5. the trains, if they have to, always stop for 10 minutes in a station,
6. in order to simplify the system, we will consider that ony one train at a time can pass through during a connection between two stations in a given direction.
   Therefore, there can be 2 trains at the same time during a connection, with each going in the opposite direction. In other words, two trains can pass each other, but not follow one another. For instance, if a train connects Paris to Rennes between 9:07 and 10:43, in consequence no other train can connect Paris to Rennes between 9:07 and 10:43. But a train can connect Rennes to Paris between 9:07 and 10:43.

Mandatory section

Instructions

You must create a program that, once launched, will expect one of the following commands on the standard input:

- create
- delete
- list
- quit

Create command
The `create` command takes the following arguments as parameter:

- the type of train,
- the depature date in dd-mm-yyyy format,
- the departure time in hh:mm format,
- the list of stations visited, in order and separated by a comma (for example, the "Paris,Madrid,London" list means that the train leaves from Paris, stops in Madrid and terminates in London, whatever illogical it may be in real life). The list must contain 2 cities at least.

This command creates a trip that leaves from the first station at the date and time of departure. An index is associated to the trip. This index combines the type of train and a 4 digits number (separated by a space, example : TGV 4034). Then it's going to calculate the dates and times of the stops in the other stations, according to train's average speed and the distance between said stations. The trains' average speeds and the distances between stations are provided further on in this document.

Another constraint to take into account is the time a train stops in a station.
To simplify things, the time a train stops will be the same in all of the stations: 10 minutes.
Moreover, stations have an infinite number of plateforms.
If the command executes correctly (no conflicts with other trips), then the function displays a message as in the following example:

```
Trip created: TGV 4234
```

If the command cannot create a trip, it displays a message containing the conflict, as in the following example.

```
Trip not created: conflict with TGV 4234
```

Here's a command example that creates a trip:

Terminal

```
~/B-PAV-360> ./trainmanager
create TGV 10-02-2017 09:34 Paris,Lyon,Marseille
Trip created: TGV 4234
```

delete command
The `delete` command takes the index of a trip and deletes the trip. Example of index : TGV3847, Eurostar2398...

List command
The `list` command doesn't take an argument and displays the list of known trips on the standard output. The format is the same as for create (the example below is only for format).

Terminal

```
~/B-PAV-360> ./trainmanager
create TGV 10-02-2017 09:34 Paris,Lyon,Marseille
Trip created:  TGV 9344
create Eurostar 25-04-2017 07:04 Brussels,Lille,London
Trip created:  Eurostar 6685
list
Eurostar 6685
Brussels (,) (25-04-2017,07:04)
Lille (25-04-2017,07:43) (25-04-2017,07:53)
London (25-04-2017,09:33) (,)
TGV 9344
Paris (,) (10-02-2017,09:34)
Lyon (10-02-2017,11:25) (10-02-2017,11:35)
Marseille (10-02-2017,12:59) (,)
```

quit command
The `quit` command simply close the program (bonus: you can save created trips on file).

Modules and Functors

A train management system is going to have to manage different types of trains (TGV, Eurostar, Thalys,...) with different constraints. The TGV trains connect the big cities, while the Eurostar only connects Paris, Lille, London and Brussels.
You should also define a "journey" or a "trip" by taking the following into account: the type of train, the stations visited and the departure and arrival times.

In order to depict your network and handle it more easily, you are required to use the modules, sub-modules and especially the functors, which allow you to define a module from another module.

The functors could be practical for creating a Trip module, depending on the Train Module that is given as parameter... It's up to you to define your modules' architecture, but you must be able to explain it and you must use the `functor` keyword.

Data

Here are the trains' average speed:

```
TGV       : 230 km/h
Thalys    : 210 km/h
Eurostar  : 160 km/h
```

Stations served by the train (the cities are written without accents).
Don't take offense. All of the cities that are actually served by the trains aren't necessarily on the list. I would remind you that this is an exercise:

```
TGV       : Brest, Le Havre, Lille, Paris, Strasbourg, Nancy, Dijon, Lyon, Nice, Marseille,
            Montpellier, Perpignan, Bordeaux, Nantes, Avignon, Rennes, Biarritz, Toulouse, Le Mans,
Thalys    : Paris, Lille, Liege, Brussels, Amsterdam, Cologne, Essen
Eurostar  : Paris, London, Brussels, Lille
```

The distances are the following:

```
    Paris – Lyon : 427 km
    Dijon – Lyon : 192 km
    Paris – Lille : 225 km
    Paris – Nancy : 327 km
    Dijon – Nancy : 226 km
    Brest – Rennes : 248 km
    Lille – London : 269 km
    Liege – Cologne : 118 km
    Le Mans – Paris : 201 km
    Cologne – Essen : 81 km
    Lyon – Marseille : 325 km
    Brussels – Liege : 104 km
    Paris – Le Havre : 230 km
    Rennes – Le Mans : 163 km
    Le Mans – Nantes : 183 km
    Paris – Bordeaux : 568 km
    Lille – Brussels : 106 km
    Nancy – Strasbourg : 149 km
    Paris – Strasbourg : 449 km
    Dijon – Strasbourg : 309 km
    Toulouse – Bordeaux : 256 km
    Brussels – Amsterdam : 211 km
    Montpellier – Toulouse : 248 km
    Marseille – Montpellier : 176 km
```