B4 - Functional Programming

B-FUN-400

Bootstrap

Wolfram

1.0

Part I - Type and data structures

Haksell features several ways to define your own types and data structures.

Configuration

Write a data structure called **Conf** suitable to store all the parameters of your program.
Each parameter must be represented with the most suitable type, and abscence of definition should not be encoded as magic values but using **Maybe**.

The **Day 3** of the **Paradigm Seminar** may be helpful here

In an ideal world, your data structure should only be able to represent valid states for the program.

Default Configuration

Write a default configuration object holding the default values for the options for which they exist, and Nothing otherwise.
It's prototype must be:

```
defaultConf :: Conf
```

Set all options from a list

Write a function with the folowing prototype:

```
getOpts :: Conf -> [String] -> Maybe Conf
```

This function takes a Conf object and a list of arguments, as provided by getArgs, and returns a new Conf object wrapped in a Maybe, or Nothing if an error occured.

PART II - STACK

- Install Stack, the most used platform for developping in Haskell, and find out details about this tool.
- Create a new project with stack for the **Wolfram**.
- Edit your stack project to use the recommanded resolver: **lts-20.11**.
- write a makefile which copies your binary to the root of your project.

The automatic test system expects a resolver identified as a **version tag**, not as a full url

'stack build' puts your executable in a directory that is **system-dependent**, which you may want to copy.
A useful command to learn this path in a system-**independent** way is:
`stack path --local-install-root`.

The automatic test system expects to find the file stack.yaml of your project at the root of your repository