INGI1131 Practical Exercises
Lab 1 extra: for hard rock programmers

Functors

In this extra series of exercises we will learn how to make Mozart programs that can be compiled and run without the OPI. For that purpose we will use *functors*. An Oz functor is equivalent to a module in other languages. For a complete description, look at the Application Development section in the Mozart Documentation. It can be found on the following url: `http://www.mozart-oz.org/documentation/apptut/node3.html#chapter.development`

Here is an example of a functor printing `Hello Nurse` in two different ways.

```
functor
import
   Application
   System
define
   {System.show 'Hello Nurse'}
   {System.showInfo "Hello Nurse"}
   {Application.exit 0}
end
```

Compiling and executing a functor

Assuming you put the code of the above functor in a file named foo.oz, it can be compiled as follows in a terminal:

```
ozc -x foo.oz
```

The program `ozc` is the oz compiler. The option `-x` indicates that we are creating an executable file. That line will generate an executable file called `foo`. To run the program execute the following in a terminal:

```
./foo
```

Try making the `foo` functor, compile it and run it.

Defining variable in a functor

Let us rewrite the `foo` functor using a variable. Here we will use `define ... in ...`. Note that it is `define`, not `declare`.

```
functor
import
   Application
   System
define
   Hello = 'Hello Nurse'
in
   {System.show Hello}
   {System.showInfo Hello}
   {Application.exit 0}
end
```

1. Compile and execute this new functor.

2. From Lab01, take exercise 4, and build it into a functor. Compile and execute it.