Pamphlet 7, INF222, Spring 2021

7.1 Documentation strings

A calculator with many primitive functions, will have several functions that are unfamiliar for many users. Therefore it will be useful if the declaration of the primitive functions can be extended with documentation.

In many languages special documentation comments may be associated with declarations. These may then be extracted by documentation generators, like Javadoc, Doxygen or Haddock, which automatically provide documentation of software systems. Also many integrated development systems (IDEs), like Eclipse, Emacs, Microsoft's Visual Studio or Apple's Xcode, automatically display the documentation comment when the user navigates the code.

Documentation for Javadoc (for Java) and Doxygen (for C/C++ and other languages) uses the comment format `/** .. */` (multiline), while Haddock (for Haskell) uses the comment format `-- | ..` (single lines) or `{- | .. -}` (multiline).

In the calculator setting we want to add documentation to help the user of the calculator understand the primitive functions. We can add a documentation string to the declaration of primitive functions. This documentation string can then be displayed together with the declarations in the welcome message from `main`.

```
type Primitive = (String,[String],String,String)
```

Though this declaration of `Primitive` is technically sufficient for our purpose, it provides very little guidance in which string is used for what. We can improve the documentation aspect of the Haskell declarations by introducing extra types.

```
-- | A primitive function declaration defines
-- a function name, a list of parameter types, a return type, and a documentation string.
-- A documentation string can have embedded new lines.
type Primitive = (FunctionName,[TypeName],TypeName,DocString)

-- | Specific type names to differentiate between distinct purposes: function names.
type FunctionName = String
-- | Specific type names to differentiate between distinct purposes: type names.
type TypeName = String
-- | Specific type names to differentiate between distinct purposes: documentation strings.
type DocString = String
```

Using this format, we may declare the least common multiplier function:

```
      ("Lcm", ["Integer","Integer"], "Integer",
       "Least common multiplier of two integers,\ne.g., Lcm [10,15]==30")
```

Note the use of the embedded new line characters, written `"\n"`, in the documentation string, making it possible to write multiline documentation.

We may then print the declaration using the Haskell documentation convention. The following could be an excerpt of the output from the start of the main function listing the primitive functions.

```
-- | Least common multiplier of two integers,
-- | e.g., Lcm [10,15]==30
Lcm :: Integer, Integer -> Integer
```

7.2 Tasks

7.2.1 Add documentation strings

Do the suggested modification to `type Primitive` in `CalculatorExternalPrimitivesAST`. This will induce several errors in the code in module `Calculator8IntegerPrimitives`.

The important fix is to `integeroperations` where documentation strings should be added.

The other errors in `Calculator8IntegerPrimitives` can be fixed by changing the pattern match triple `(fn, args, res)` to a quadruple `(fn, args, res, doc)`.

7.2.2 Present the documentation strings

Rewrite the `main` function to print the documentation strings, formatted as suggested above.