Lab 7

Functional Programming (ITI0212)

2023-03-17

This week we learned how to do monadic I/O in typed *purely functional* programming languages. Unlike *imperative* programming languages these do not have a syntactic class of *statements*. Instead, some *expressions* represent *computations*, which are instructions to the run-time system to perform various *actions*. These are distinguished in the type system by being elements of `IO` types.

We can build up compound computations from simpler ones using two *monadic combinators*, `pure : a -> IO a`, which produces a trivial computation, and `(>>=) : IO a -> (a -> IO b)-> IO b`, which sequences computations by running the first and passing the resulting value to the next.

There is syntactic sugar called *`do`-notation*, in which a sequence of computations can be written to resemble a block of statements in an imperative programming language. This can be convenient, but it is important to understand that it is merely a syntactic transformation: purely functional programming languages do not have statements.

Task 1
Write a computation,

```
concat_strings : IO String
```

which when run, it concatenates two strings. Write two implementations, one using the *`do`-notation* and the other using the sequencing operator `(>>=)`. For example,

```
Lab7>:exec concat_strings >>= printLn
Please enter the first sentence:
Hello
Please enter the second sentence:
there!
"Hello there!"
```

Task 2
Write a function,

```
add_after : Integer -> IO (Maybe Integer)
```

, which takes an integer as an input, and returns a computation which when run asks the user to input another number and adds them together. In case the user does not provide a number, then the function should return `Nothing`. For example,

```
Lab7>:exec add_after 10 >>= printLn
Please enter a number:
7
Just 17
Lab7>:exec add_after 10 >>= printLn
Please enter a number:
no
```

```
Nothing
```

Task 3
Write a computation

```
count_words  :  IO Unit
```

which when run asks the user to enter some text on a line and prints on the terminal the number of words that the user wrote. For example,

```
Lab7>:exec count_words
Enter some text:
Hello world!
You typed 2 words.

Leb7>:exec count_words
Enter some text:
Some numbers- 5, 3- and words separated by space.
You typed 9 words.
```

Hint: You can use the `words` function form `Data.String`.

Task 4
Write a computation,

```
get_lines : IO(List String)
```

that reads lines of user input until the user enters "done", and returns the lines as a `List String`. For example,

```
Lab7>:exec get_lines >>= printLn
Please enter a sentence:
Some first sentence
Please enter a sentence:
Hello there
Please enter a sentence:
done
["Some first sentence", "Hello there"]
```

Task 5
Write a (similar) computation as the one before,

```
get_only_ints : IO (List Integer)
```

, which repeatedly asks the user to enter an integer until the user types "done" and returns a list containing the integers that were given.

```
Lab7>:exec get_only_ints >>= printLn
Please enter an integer or "done":
4
Please enter an integer or "done":
no
Please enter an integer or "done":
6
Please enter an integer or "done":
done should be the only word on a line to end this
Please enter an integer or "done":
7
```

```
Please enter an integer or "done":
done
[4, 6, 7]
```

Task 6
Warning: In this exercise you will use a function to write to a file. Please make sure that you are not overwriting an important file on your computer!

Write a computation

```
dictate : IO ()
```

and compile it to obtain an executable (here also named dictate). When run, dictate should read lines until the user enters “done”, then prompt the user for the name of a file to store those lines in. A message relaying the success of failure of this operation should be printed before the program exits. For example:

```
Lab7>./dictate
Please enter a sentence:
Hello!
Please enter a sentence:
I want to test this function.
Please enter a sentence:
done
Enter the location:
dictate.txt
Success!
>cat dictate.txt
Hello! I want to test this function.

Lab7>./dictate
Please enter a sentence:
Testing this
Please enter a sentence:
on a bad example.
Please enter a sentence:
done
Enter the location:
bad_directory/dictate.txt
Failed to write to file.
```

Note: You will need to `import System.File`.

Hint: You may want to use the `get_lines` function you defined before as well as the `unwords` and `writeFile` functions.