Functional and Logic Programming

Home Assignment 5

Due: 25 June 2022 - 23:59

Instructions

- Please create a source file called ***hw5.erl*** and put all the answers there.
  The file should start with a comment, which contains your **full name** (in English) and **ID.**

  % Montgomery Burns
  % 15926535

- **Important:** Please name the Erlang module **hw5**

  -module(hw5).

- **Important 2:** Please make sure your code compiles without errors or warnings.
- **Important 3:** Feel free to use standard library functions like lists:map, lists:sort or io:format.

Question 1 – 10%

Define a function create_indices/1, which takes a non-negative integer N and returns the list [0,1,2,….,N].

Question 2 – 25%

Define a function ex_map/2, which works like the standard map/2 but takes a different function as the first parameter. ex_map/2 will use a function having **two parameters**. The first is the list's item and the 2nd is its index in the list.

Note: please provide a **non-recursive** solution. You can use create_indices/1 in your solution.

Example

hw5:ex_map(fun(X, Y) -> X == Y end, [0,2,2,2]) should return [true,false,true,false]

Question 3 – 15%

Define the function fizzbuzz/1. The function takes a non-negative integer N and returns a list of length N. We index the output starting with 1 (Not 0). The outputted list will be constructed as follows:

- If the index of the item is divisible by 3 then the output will be the atom fizz.
- If the index of the item is divisible by 5 then the output will be the atom buzz.
- If the index of the item is divisible by both 3 and 5, then the output will be the atom fizzbuzz.
- In all other cases, the output will be the index.

See https://en.wikipedia.org/wiki/Fizz_buzz for more details.

Note: please provide a **non-recursive** solution. You can use create_indices/1 in your solution.

Example: If N is 15 the output will be:

[1,2,fizz,4,buzz,fizz,7,8,fizz,buzz,11,fizz,13,14,fizzbuzz]

Question 4 – 50%

Define a function concurrent_filter/2 which works exactly like lists.filter. The only difference is that concurrent_filter/2 will run all the predicate operations concurrently.

Example:

```
Odd = fun(X)-> ( X rem 2 ) /= 0 end .
```

hw5:concurrent_filter(Odd, [1,2,3,4,5]) will run the function Odd in 5 different processes, will collect the results from all of them and return [1,3,5].