Exam of Programming Languages

15 December 2014

Exercise Erlang: Distributed Sieves.

```
-module(client).
-export([is_prime/1, close/0]).

is_prime(N) -> send_msg({new, N, self()}).
close() -> send_msg({quit, self()}).

send_msg(M) ->
  {controller, sif@surtur} ! M,
  receive
    {result, R} ->  io:format("~p~n", [R])
  end.
```

```
-module(controller).
-export([start/1]).

start(N) -> register(controller, spawn(fun() -> init_ring(N) end)).

init_ring(N) ->
  loop(N, connect([spawn_link(sieve, init, [X]) || X <- lists:seq(2,N),
      (length([Y||Y<-lists:seq(2, trunc(math:sqrt(X))), ((X rem Y) == 0)])==0)])).

connect(L=[H|TL]) -> connect(H, L, TL++[H]).

connect(Pid, [Pid1|[]], [Pid2|[]]) ->
   Pid1! {who, self()},
   receive
      {who, Max} -> Pid1 !{Pid, Pid2}, loop(Max, Pid)
   end;
connect(Pid, [Pid1|TL1], [Pid2 |TL2]) -> Pid1 !{Pid, Pid2}, connect(Pid, TL1, TL2).

loop(Max, Head) ->
  receive
    {new, N, From} -> io:format("You asked for: ~p~n", [N]),
       RootN = trunc(math:sqrt(N)),
       if
         (RootN < Max) ->
           Head! {new, N},
           receive {res, V} ->
             From!{result,
               lists:flatten(io_lib:format("is ~p prime? ~p",[N,V]))}
           end,
           loop(Max, Head);
         true ->
           From!{result,lists:flatten(
               io_lib:format("~p is uncheckable, too big value.",[N]))},
           loop(Max, Head)
       end;
    {quit, From} ->
       io:format("I'm closing ...~n"), From ! {result, "The service is closed!!!"}
  end.
```

```
-module(sieve).
-export([init/1]).

init(N) ->
  receive
    {who, From} -> From ! {who, N}, init(N);
    {Gate, To} -> loop(Gate, To, N)
  end.

loop(Gate, To, N) ->
  receive
    {new, N1} -> Gate ! {pass, N1}, loop(Gate, To, N);
    {pass, N1} ->
       RootN1 = trunc(math:sqrt(N1)),
       if
```

```
        (N > RootN1) -> Gate ! {res, true}, loop(Gate, To, N);
        ((N1 rem N) == 0) -> Gate ! {res, false}, loop(Gate, To, N);
        true -> To ! {pass, N1}, loop(Gate, To, N)
      end;
   {res, V} -> controller ! {res, V},  loop(Gate, To, N);
   Other -> io:format("unrecognized message:- ~p~n", [Other]), loop(Gate, To, N)
 end.
```