**ERLANG**

-module(exercise\_01).

-compile(export\_all).

% -export([

% hello\_world/0,

% hello\_world/1,

% greet/1

% ]).

hello\_world() ->

"Hello world".

hello\_world(Name) ->

"Hello " ++ Name.

greet(Name) when list(Name) ->

"Ciao " ++ Name ++ "!";

greet(\_) ->

"Sorry, insert a string!!!".

greet\_adult(Name, Age) ->

if

Age >= 18 -> "Hello " ++ Name;

Age < 18 -> "Too small"

end.

is\_adult(Age) when Age >= 18 -> true;

is\_adult(\_) -> false.

greet\_adult\_case(Name, Age) ->

case is\_adult(Age) of

true -> "Hello " ++ Name;

false -> "Smol"

end.

how\_long([]) -> 0;

how\_long([\_ | Xs]) -> 1 + how\_long(Xs).

% Rock, Paper, Scissors

rps\_player() ->

Pid = spawn(fun() -> rps\_cpu() end),

io:format("RPS Pid = ~p~n", [Pid]),

Pid ! {self(), rock},

rps\_win\_loop().

rps\_win\_loop() ->

receive

win -> io:format("Hoorah!~n");

Result -> io:format("~p~n", [Result])

after 5000 -> exit(ok)

end.

rps\_win(PlayerChoice, CPUChoice) ->

case {PlayerChoice, CPUChoice} of

{rock, rock} -> tie;

{rock, scissors} -> win;

{rock, paper} -> loose

end.

rps\_cpu() ->

io:format("[CPU] Ready!~n"),

RPS = [rock, paper, scissors],

CPUChoice = lists:nth(rand:uniform(2), RPS),

receive

{PlayerPid, PlayerChoice} -> PlayerPid ! rps\_win(PlayerChoice, CPUChoice)

end,

rps\_cpu().

% SPAWN\_LINK

% Crashing child

bad\_proc() ->

io:format("I'm going to crash.~n"),

4/0.

test\_spawn\_link() ->

process\_flag(trap\_exit, true),

Pid = spawn\_link(fun() -> bad\_proc() end),

receive

Err -> io:format("Received message from child: ~p~n", [Err])

end.

% Dying parent

dying\_parent(N) ->

spawn\_link(fun() -> child(N) end),

io:format("Parent is gonna die~n"),

receive

after 3000 ->

io:format("Parent is dead~n"),

exit(ok)

end.

child(0) -> exit(ok);

child(N) ->

receive

after 1000 -> io:format("Hey!~n")

end,

child(N - 1).

% MONITOR

dying\_parent\_m() ->

% because we want to send a child a Pid of a parent

MyPid = self(),

Pid = spawn(fun() -> child\_m(MyPid) end),

receive

after 1000 ->

io:format("Parent died~n"),

exit(died)

end.

child\_m(ParentPid) ->

io:format("Monitoring parent~n"),

monitor(process, ParentPid),

receive

Any -> io:format("Parent's dead!~n~p~n", [Any])

after 2000 -> ok

end.