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.
```