Step 2.10 ASSIGNMENT: Supervision

In this assignment your task is to complete a supervision module for a system built of two processes: one sends messages, and the other echoes them.

The basic system

The system consists of two modules. One `talk.erl` sends messages to the other `echo.erl`, which echoes them back. The code in these modules is instrumented so that it prints out diagnostic messages to the Erlang shell as it operates.
The code in `talk.erl` also includes a call to `timer:sleep/1` to ensure that messages are sent at the rate of about two per second. The code is shown here, and can be downloaded from Step 2.9 of the course.

```
-module(talk).
-export([worker/0]).

worker() ->
    work(0).

work(N) ->
    Msg = {self(),N},
    echo!Msg,
    io:format("~w sent.~n",[Msg]),
    receive
        _Reply ->
            timer:sleep(500),
            work(N+1)
    end.
```

```
-module(echo).
-export([listener/0]).

listener() ->
    receive
        {Pid,M} ->
            io:format("~w echoed.~n",[M]),
            Pid!M,
            listener()
    end.
```

Using the system

You will see that the function `talk:worker` sends messages to the registered process `echo`, which returns them to the Pid contained in the incoming message. This code sets up the system

```
1> register(echo,spawn(echo,listener,[])).
true
2> Pid=spawn(talk,worker,[]).
<0.60.0>
{<0.60.0>,0} sent.
0 echoed.
{<0.60.0>,1} sent.
1 echoed.
```

To observe how the system behaves when one of the processes is killed we use `exit(Pid,kill)` to kill the worker, whereas for the echo process we use `whereis` to look up its Pid: `exit(whereis(echo),kill)`.

Experiment with calling these functions, and with restarting the killed process. In particular, what happens to the processes when you kill the `echo` process?

Include your answers as comments in your supervisor module.

Adding a supervisor

You are to add a supervisor to this system; here's a skeleton to complete:

```
-module(super).
-export([super/0]).

super() ->
    process_flag(trap_exit, true),
    E = spawn_link(echo,listener,[]),
    register(echo,E),
    io:format("echo spawned.~n"),
    T = spawn_link(talk,worker,[]),
    register(talk,T),
    io:format("worked spawned as Pid ~w.~n",[whereis(talk)]),
    loop(E,T).

loop(E,T) ->
    … to complete …
```

As you can see in the code, the supervisor activates `trap_exit` to enable processing exit signals it receives as messages, and then the two processes are spawned, linked and registered. We register both processes so that it's possible for you to kill them by looking up their Pids using `whereis`.

You need to complete to code for `loop/2`, which takes the (current) Pids of the two processes as arguments.

What happens when you kill either of the processes? Add a 1 second delay between receiving an EXIT message from the echo process and restarting it: after this change, does the behaviour of the system after the echo process is killed change? Explain your answer.

Please include your written answers as comments in the supervisor module.

Assessing the assignment

- Ensure that the solution is correct.
- Ensure that the solution is readable; e.g. include comments that explain how the solution works, and indicate any particular aspects that need explanation.
- Use comments to describe the behaviour of the system when either of the processes is killed.
