**To be run in the Erlang Shell**

F = fun() -> 2 + 2 end.

spawn(F).

self().

spawn(fun() -> io:format("~p~n", [2 + 2]) end).

G = fun(X) -> timer:sleep(10), io:format("~p~n", [X]) end.

[spawn(fun() -> G(X) end) || X <-lists:seq(1, 10)].

G = fun(X, S) -> timer:sleep(S), io:format("~p : ~p~n", [X, S]) end.

[spawn(fun() -> G(X , rand:uniform(2000)) end) || X <-lists:seq(1, 10)].

spawn(concurrent, from\_module, [10]).

self() ! hello.

flush().

Greet = spawn(concurrent, listener, []).

Greet ! hello.

Greet ! hi.

Greet2 = spawn(concurrent, listener2, []).

Greet2 ! {self(), hello}.

self().

flush().

Greet3 = spawn(concurrent, listener3, []).

Greet3 ! {self(), hello}.

Greet3 ! {self(), hi}.

Greet3 ! {self(), other}.

Greet3 ! {self(), hi}.

flush().

flush().

E = concurrent:createLogger("ERROR").

E(["Hello World"]).

D = concurrent:createLogger("DEBUG").

D(["This shouldn't happen!"]).

End = spawn(concurrent, terminator, [concurrent:createLogger(the\_end)]).

End.

End ! “Hello”.

Mid1 = spawn(concurrent, middle, [concurrent:createLogger(middle1), End]).

Mid1.

Mid1 ! "Hello".

Mid2 = spawn(concurrent, middle, [concurrent:createLogger(middle2), Mid1]).

Mid2.

Mid2 ! “Hello”.

MidF = spawn(concurrent, middleWare, [concurrent:createLogger(mw), fun (X) -> X \* X end, End]).

MidF ! 5.

MidF2 = spawn(concurrent, middleWare, [concurrent:createLogger(mw2), fun (X) -> X \* X end, MidF]).

MidF2 ! 5.

P = concurrent:go().

P ! hello.