3 Concurrency Oriented Programming [12 points]

1. Study the following Erlang code:

```
foo() ->
    Self = self(),
    spawn(fun() -> Self ! a end),
    spawn(fun() -> Self ! b end),
    receive A -> receive B -> {A,B} end end.
```

   What are the possible results that `foo()` can return? **1 points**

2. Erlang does not provide locks to protect shared data from simultaneous modification by two or more concurrent processes. What prevents Erlang processes from corrupting shared data? **1 points**

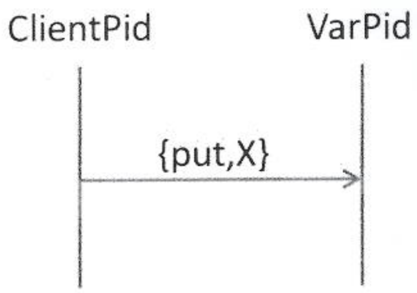
3. Shared variables can be implemented in Erlang as processes. Study the following code, which defines a process `variable(X)` that is intended to behave like a variable containing the value `X`.

```
variable(X) ->
    spawn(fun() -> var(X) end).

var(X) ->
    receive
        {put,Y} ->
            ...
    end.

putvar(VarPid,X) ->
    VarPid ! {put,X}.
```

   The following diagram illustrates the messages passed by a call of `putvar(VarPid,X)`, which is intended to set the value of the variable to `X`.


ClientPid
VarPid
{put,X}