Paradigmas de la Programación
Recuperatorio del Segundo Parcial

23 de Junio de 2022

Apellido y Nombre: _____

1	
2	
3	

4	
5	

1. **[10 pt.]** ¿Qué características de los lenguajes de scripting se pueden observar en el siguiente programa? Siempre que sea posible, cite los fragmentos específicos de código en los que se observan las propiedades que mencione.

```
tell application "Finder"
    set passAns to "app123"
    set userAns to "John"
    if the text returned of (display dialog "Username" default answer "") is userAns then
        display dialog "Correct" buttons {"Continue"} default button 1
        if the text returned of (display dialog "Username : John" & return & "Password" default answer ""
buttons {"Continue"} default button 1 with hidden answer) is passAns then
            display dialog "Access granted" buttons {"OK"} default button 1
        else
            display dialog "Incorrect password" buttons {"OK"} default button 1
        end if
    else
        display dialog "Incorrect username" buttons {"OK"} default button 1
    end if
end tell
```

2. **[25 pt.]** El siguiente programa está escrito en Erlang. Encuentre en el texto del programa por lo menos dos características propias del paradigma de actores y relaciónelas con lo que conoce de Akka.

```
% Create a process and invoke the function web:start_server(Port, MaxConnections)
ServerProcess = spawn(web, start_server, [Port, MaxConnections]),

% Create a remote process and invoke the function
% web:start_server(Port, MaxConnections) on machine RemoteNode
RemoteProcess = spawn(RemoteNode, web, start_server, [Port, MaxConnections]),

% Send a message to ServerProcess (asynchronously). The message consists of a tuple
% with the atom "pause" and the number "10".
ServerProcess ! {pause, 10},

% Receive messages sent to this process
receive
        a_message -> do_something;
        {data, DataContent} -> handle(DataContent);
        {hello, Text} -> io:format("Got hello message: ~s", [Text]);
        {goodbye, Text} -> io:format("Got goodbye message: ~s", [Text])
end.
```

3. **[10 pt.]** Dada la siguiente base de conocimiento, ¿qué va a contestar el intérprete si le preguntamos `digiriendo(rana,mosca).`, y cómo va a llegar a su respuesta?

```
digiriendo(X,Y) :- comio(X,Y).
digiriendo(X,Y) :-
        comio(X,Z),
        digiriendo(Z,Y).

comio(mosquito,sangre(juan)).
comio(rana,mosquito).
comio(gaviota,rana).
```

4. **[30 pt.]** En el siguiente texto nos explican una característica muy interesante de Elm. Esta característica, ¿contribuye negativamente o positivamente a la seguridad del lenguaje? ¿por qué? ¿y a la robustez del lenguaje? ¿Qué mecanismos de programación defensiva podríamos aplicar en otro lenguaje para tener un comportamiento parecido al comportamiento que presenta Elm en este programa?

One of the guarantees of Elm is that you will not see runtime errors in practice. This is partly because Elm treats errors as data. Rather than crashing, we model the possibility of failure explicitly with custom types. For example, say you want to turn user input into an age. You might create a custom type like this:

```
type MaybeAge
  = Age Int
  | InvalidInput

toAge : String -> MaybeAge
toAge userInput =
  ...

-- toAge "24" == Age 24
-- toAge "99" == Age 99
-- toAge "ZZ" == InvalidInput
```

Instead of crashing on bad input, we say explicitly that the result may be an Age 24 or an InvalidInput. No matter what input we get, we always produce one of these two variants. From there, we use pattern matching which will ensure that both possibilities are accounted for. No crashing!

5. **[25 pt.]** El siguiente texto explica los conceptos de *hot spot* y *frozen spot* en frameworks. También nos explica cómo funcionan los frameworks con orientación a objetos. Cuando instanciamos una aplicación con un framework basado en objetos, terminamos teniendo un sistema de objetos específico para esa aplicación, basado en el sistema de objetos provisto por el framework. Basándose en los conceptos de *concreto* y *abstracto* y *composición* y *subclases* que usa el texto, explique qué componentes de este sistema de objetos se podrían considerar *frozen spots* y cuáles se podrían considerar *hot spots*.

> *Software frameworks consist of frozen spots and hot spots. Frozen spots define the overall architecture of a software system, that is to say its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. Hot spots represent those parts where the programmers using the framework add their own code to add the functionality specific to their own project.*
>
> *In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes.*