Paradigmas de la Programación
Recuperatorio del Segundo Parcial

22 de Junio de 2023

Apellido y Nombre: _____

1. ¿Qué tipo de lenguaje de scripting observamos en el siguiente ejemplo? **[5 pt.]** Nombren por lo menos 2 características de este tipo de lenguajes de scripting **[10 pt.]**

```
1  // Player entity variables
2  entity player;
3  float playerSpeed = 100.0;
4
5  void main() {
6      // Initialize player entity
7      player = spawn();
8      setmodel(player, "player.mdl");
9      setsize(player, Vector(-16, -16, 0), Vector(16, 16, 72));
10     setorigin(player, Vector(0, 0, 0));
11     player.think = Player_Think;
12     player.movetype = MOVETYPE_WALK;
13     player.solid = SOLID_BBOX;
14
15     // Start the game loop
16     while (1) {
17         // Process keyboard input
18         if (key_down(K_LEFTARROW)) {
19             player.velocity_y = -playerSpeed;
20         } else if (key_down(K_RIGHTARROW)) {
21             player.velocity_y = playerSpeed;
22         } else {
23             player.velocity_y = 0;
24         }
25
26         // Update the player entity
27         setorigin(player, player.origin + player.velocity * frametime);
28
29         // Update the game state
30         progs_run();
31     }
32 }
```

2. Identifique en el siguiente programa *hot spots*, con *frozen spots* **[10 pt.]**, y explique cómo se instancia el framework a través de herencia de clases **[5 pt.]**

```
import React from 'react';

class Button extends React.Component {
  handleClick() {
    console.log('Button clicked');
  }

  render() {
    return (
      <button onClick={this.handleClick}>Click Me</button>
    );
  }
}

class CustomButton extends Button {
  handleClick() {
    super.handleClick(); // Call the parent class method
    console.log('CustomButton clicked');
  }

  render() {
    return (
      <div>
        <button onClick={this.handleClick}>Custom Button</button>
      </div>
    );
  }
}

export default CustomButton;
```

3. En el siguiente programa en Prolog, en los lugares del código que ocupan las letras A, y C debería haber alguna de las variables que ya se encuentran en la regla. Escriban a cuál de las variables se corresponde cada una de las letras **[10 pt.]** y expliquen la función de la variable B **[5 pt.]**

```
package(p1, location(a)).
package(p2, location(b)).
package(p3, location(c)).
package(p4, location(d)).

transport(truck, a, b).
transport(plane, b, c).
transport(train, c, d).
transport(ship, d, a).

connected(X, Y) :-
```

```
    transport(_, X, Y).
connected(X, Y) :-
    transport(_, Y, X).

reachable(Package, Destination) :-
    package(A, B),
    find_route(B, C, [Package]).

find_route(Source, Destination, Visited) :-
    connected(Source, Destination),
    \+ member(Destination, Visited).

find_route(Source, Destination, Visited) :-
    connected(Source, Interim),
    \+ member(Interim, Visited),
    find_route(Interim, Destination, [Interim|Visited]).
```

4. En el siguiente fragmento de código se está aplicando una estrategia de programación defensiva. Identifíquela **[10 pt.]** y describa cómo funciona **[5 pt.]**

```
def divide_numbers(dividend, divisor):
    if divisor == 0:
        raise ValueError("Cannot divide by zero.")
    result = dividend / divisor
    return result

def get_user_input():
    try:
        dividend = int(input("Enter the dividend: "))
        divisor = int(input("Enter the divisor: "))
        result = divide_numbers(dividend, divisor)
        print(f"The result of the division is: {result}")
    except ValueError as e:
        print("Invalid input:", e)
    except Exception as e:
        print("An error occurred:", e)

get_user_input()
```

5. En el siguiente programa, identifique los mecanismos por los cuales se garantiza concurrencia declarativa **[10 pt.]** y explique cómo funcionan esos mecanismos para evitar la existencia de una sección crítica **[5 pt.]**

```
from actor import Actor, ActorSystem

# Actor implementation
class Counter(Actor):
    def __init__(self):
        self.count = 0

    def receive(self, message, sender):
        if message == 'increment':
            self.count += 1
            print(f'Counter: {self.count}')
        elif message == 'get_count':
            sender.send(self.count)

# Create an actor system
system = ActorSystem()

# Create an instance of the Counter actor
counter = system.create_actor(Counter)

# Send messages to the Counter actor
counter.send('increment')
counter.send('increment')
counter.send('get_count')

# Wait for the response
response = system.receive()

# Print the count
print(f'Final count: {response}')

# Shut down the actor system
system.shutdown()
```