Paradigmas de la Programación – Examen Final

9 de Agosto de 2023

Apellido y Nombre: _____

1. **[10 pt.]** El siguiente texto explica lo que es un *thunk*.

   > *A simple implementation of "call by name" might substitute the code of an argument expression for each appearance of the corresponding parameter in the subroutine, known as a "thunk".*

   Escriba el *thunk* resultante de compilar `f(array[i])` en el siguiente programa, donde los argumentos se pasan mediante la estrategia *call-by-name*:

```
1  int i;
2  int array[3] = { 0, 1, 2 };
3
4  i = 0;
5  f(array[i]);
6
7  int f(int j)
8  {
9      int k = j;
10     i = 2;
11     k = j;
12 }
```

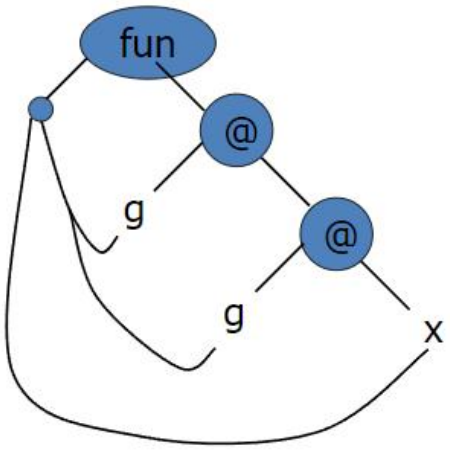
2. **[20 pt.]** ¿Qué imprime el siguiente programa con alcance estático, y qué imprime con alcance dinámico? Dibuje los diferentes estados por los que pasa la pila de ejecución.

```
1  def foo():
2      x = 10
3
4      def bar():
5          print("valor de x:", x)
6
7      local x
8      x = 20
9      bar()
10
11 foo()
```

3. **[10 pt.]** Al ejecutar el siguiente programa, el archivo `archivo.txt` no existe. Explique verbalmente qué va sucediendo durante esta ejecución, ayudándose de diagramas de la pila de ejecución cuando lo considere necesario. No es necesario representar las variables locales, control links, access links, retorno de función ni ninguna otra información que no sea relevante al proceso de manejo de excepciones.

```
1  public class EjemploExcepcionesFinally {
2      public static void main(String[] args) {
3          BufferedReader reader = null;
4
5          try {
6              reader = new BufferedReader(new FileReader("archivo.txt"));
7              String linea;
8
9              while ((linea = reader.readLine()) != null) {
10                 System.out.println(linea);
11             }
12         } catch (IOException e) {
13             System.out.println("Error al leer el archivo: " + e.getMessage());
14         } finally {
15             try {
16                 if (reader != null) {
17                     reader.close();
18                 }
19             } catch (IOException e) {
20                 System.out.println("Error al cerrar el archivo: " + e.getMessage()
21             }
22         }
23
24         System.out.println("Fin del programa");
25     }
26 }
```

4. **[10 pt.]** Escriba la expresión que se representa en este árbol de tipos y su signatura de tipos.



5. **[10 pt.]** En el siguiente programa en C++, ¿qué características tiene que tener SumArray para que el programa pueda ser considerado declarativo?

```
int values[4] = { 8, 23, 2, 4 };
int sum = SumArray(values);
RotateArrayIndices(values, -1);
```

6. **[10 pt.]** Explique cómo en el siguiente código en Java se usan los mecanismos de herencia y de interfaz para lograr el comportamiento deseado en el programa sin repetir código.

```
interface Animal {
    void sound();
}

class Mammal implements Animal {
    @Override
    public void sound() {
        System.out.println("Mammal sound");
    }
}

class Dog extends Mammal {
    @Override
    public void sound() {
        System.out.println("Dog barks");
    }
}

public class Example {
    public static void main(String[] args) {
        Animal dog = new Dog();

        dog.sound(); // Output: Dog barks
    }
}
```

7. En el siguiente programa en Prolog, identifique qué variables de la regla corresponden a las letras A y C **[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]).
```

8. 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()
```

9. ¿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.]**

```
// Player entity variables
entity player;
float playerSpeed = 100.0;

void main() {
    // Initialize player entity
```

```
    player = spawn();
    setmodel(player, "player.mdl");
    setsize(player, Vector(-16, -16, 0), Vector(16, 16, 72));
    setorigin(player, Vector(0, 0, 0));
    player.think = Player_Think;
    player.movetype = MOVETYPE_WALK;
    player.solid = SOLID_BBOX;

    // Start the game loop
    while (1) {
        // Process keyboard input
        if (key_down(K_LEFTARROW)) {
            player.velocity_y = -playerSpeed;
        } else if (key_down(K_RIGHTARROW)) {
            player.velocity_y = playerSpeed;
        } else {
            player.velocity_y = 0;
        }

        // Update the player entity
        setorigin(player, player.origin + player.velocity * frametime);

        // Update the game state
        progs_run();
    }
}
```

10. 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');
```

```
19    }
20
21    render() {
22      return (
23        <div>
24          <button onClick={this.handleClick}>Custom Button</button>
25        </div>
26      );
27    }
28  }
29
30  export default CustomButton;
```

11. 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.]**

```
1  from actor import Actor, ActorSystem
2
3  # Actor implementation
4  class Counter(Actor):
5      def __init__(self):
6          self.count = 0
7
8      def receive(self, message, sender):
9          if message == 'increment':
10             self.count += 1
11             print(f'Counter: {self.count}')
12         elif message == 'get_count':
13             sender.send(self.count)
14
15 # Create an actor system
16 system = ActorSystem()
17
18 # Create an instance of the Counter actor
19 counter = system.create_actor(Counter)
20
21 # Send messages to the Counter actor
22 counter.send('increment')
23 counter.send('increment')
24 counter.send('get_count')
25
26 # Wait for the response
27 response = system.receive()
28
29 # Print the count
30 print(f'Final count: {response}')
31
```

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

Ejercicios para Libres

1. **[-10 pt.]** En el siguiente fragmento de texto, identifique porciones con semántica concurrente y explíquelas.

```
public class Counting {
    public static void main(String[] args) throws InterruptedException {
        class Counter {
            int counter = 0;
            public void increment() { counter++; }
            public int get() { return counter; }
        }

        final Counter counter = new Counter();

        class CountingThread extends Thread {
            public void  run() {
                for (int x = 0; x < 500000; x++) {
                    counter.increment();
                }
            }
        }

        CountingThread t1 = new CountingThread();
        CountingThread t2 = new CountingThread();
        t1.start(); t2.start();
        t1.join(); t2.join();
        System.out.println(counter.get());
    }
}
```

2. **[-10 pt.]** Qué imprime este programa, asumiendo que el pasaje de parámetros es a) por valor, b) por referencia y c) por valor-resultado?

```
z : integer;
procedure p (x:integer)
  x := x+1 ;
  z := z+2;

z := 1;
p(z);
write(z)
```