Paradigmas de la Programación
Examen Final

10 de Agosto de 2022

Apellido y Nombre: _____

1		6	
2		7	
3		8	
4		9	
5		10	

1. **[10 pt.]** De estos dos programas, uno es declarativo y el otro no. Identifique en el programa no declarativo porciones del código en las que ocurren fenómenos no declarativos, y explíquelos.

```
def summation(n, term):
  total, k = 0, 1
  while k <= n:
      total, k = total + term(k), k + 1
  return total
```

```
def summation(n, term):
    if n == 0:
        return term(n)
    else:
    return term(n) + summation(n - 1, term)
```

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)
```

3. **[10 pt.]** Diagrame los estados por los que pasa la pila de ejecución al ir ejecutando el siguiente programa, y explique las diferencias entre la ejecución en un lenguaje con alcance estático y en un lenguaje con alcance dinámico.

```
a = 1
b = 2
c = 3

function test()

  local function g()
    local a = 2
    c = a + 4
  end

  local function f()
    local c = 5
    b = 4
    g()
  end

  f()
end

test()
```

4. **[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).
```

5. **[10 pt.]** Identifique en el siguiente texto una característica de seguridad y explique qué vulnerabilidad cubre y cómo.

```
org.apache.xml.security.Init.init(); //
Canonicalizer canon = Canonicalizer.getInstance(Canonicalizer.ALGO_ID_C14N_OMIT_COM
byte canonXmlBytes[] = canon.canonicalize(yourXmlBytes);
String canonXmlString = new String(canonXmlBytes);
```

6. **[10 pt.]** El siguiente programa está escrito en C++. Explique por qué `letsHear` se puede aplicar a `Cat` y `Dog`, usando los conceptos de herencia, subtipado, polimorfismo.

```
abstract class Animal {
    abstract String talk();
}

class Cat extends Animal {
    String talk() {
        return "Meow!";
    }
}

class Dog extends Animal {
    String talk() {
        return "Woof!";
    }
}

static void letsHear(final Animal a) {
    println(a.talk());
}

static void main(String[] args) {
    letsHear(new Cat());
    letsHear(new Dog());
}
```

7. **[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
```

8. **[10 pt.]** Explique verbalmente qué va sucediendo en la ejecución del siguiente programa, 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 manejo de excepciones. Explique también qué imprime el programa.

```
class Ejemplo3 {
  public static void main(String args[]){
    try{
        System.out.println("primera sentencia del bloque
   try");
        int num=45/3;
        System.out.println(num);
    }
    catch(ArrayIndexOutOfBoundsException e){

   System.out.println("ArrayIndexOutOfBoundsException");
    }
    finally{
        System.out.println("bloque finally");
    }
    System.out.println("fuera del bloque
   try-catch-finally");
  }
}
```

9. **[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());
    }
}
```

10. [**10 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.*