Paradigmas de la Programación – Primer Parcial

25 de Abril de 2019

Apellido y Nombre: _____

1. **[10 pt.]** La siguiente expresión está mal tipada:

   ```
   f (a, b) = a(b) + a
   ```

   Muestre con el árbol para inferencia de tipos dónde se encuentra el conflicto y en qué consiste. Cómo podría resolver este conflicto un lenguaje de tipado fuerte? y uno de tipado débil?

2. **[10 pt.]** El siguiente código (resumido) de un driver SCSI para Linux es spaghetti. ¿Por qué decimos que es spaghetti, cuál es la principal diferencia con el código estructurado? ¿Si el lenguaje obliga a usar código estructurado, qué estructura de datos puede usar el compilador para manejar la memoria de forma más eficiente? Reescriba el driver en pseudocódigo para que sea estructurado y no spaghetti.

```
wait_nomsg:
        if ((inb(tmport) & 0x04) != 0) {
                goto wait_nomsg;
        }
        ...
        for (n = 0; n < 0x30000; n++) {
                if ((inb(tmport) & 0x80) != 0) {
                        goto wait_io;
                }
        }
        goto TCM_SYNC;
wait_io:
        for (n = 0; n < 0x30000; n++) {
                if ((inb(tmport) & 0x81) == 0x0081) {
                        goto wait_io1;
                }
        }
        goto TCM_SYNC;
wait_io1:
        ...
TCM_SYNC:
        ...
```

3. **[10 pt.]** De estos dos fragmentos de códigos, el primero en Pascal y el segundo en C++, cuál es un ejemplo de sobrecarga y cuál es un ejemplo de polimorfismo? Justifique su respuesta.

```
function Add(x, y : Integer) : Integer;
begin
    Add := x + y
end;

function Add(s, t : String) : String;
begin
    Add := Concat(s, t)
end;
```

```
class List<T> {
    class Node<T> {
        T elem;
        Node<T> next;
    }
    Node<T> head;
    int length() { ... }
}
```

4. **[15 pt.]** En el siguiente programa en Lua, diagrame el estado en el que se encuentra la pila de ejecución al terminar de ejecutar las líneas señaladas con A, B, C y D en el texto del programa[1].

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

function test()

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

  local function f()
    local c = 5
    b = 4              ---> C
    g()
  end
                       ---> B
  f()
end
                       ---> A
test()
```

5. **[10 pt.]** En el siguiente programa, ¿encontraremos una diferencia si el alcance del lenguaje es estático o si el alcance es dinámico? ¿Qué se imprimiría en cada caso?

[1] Para ser más precisos, al terminar de ejecutar el equivalente en código máquina a esas líneas del código en Lua.

```
procedure p;
    x: integer;
    procedure q;
       begin x := x+1 end;
    procedure r;
       x: integer;
       begin x := 1; q; write(x) end;
    begin
    x:= 2;
    r
    end;
```

6. **[10 pt.]** Qué tres valores imprimiría este programa (con la expresión `write`) si el lenguaje en el que está programado tuviera pasaje de parámetros a) por valor, b) por referencia, c) por valor resultado y d) por nombre?

```
int i, A[2]
i <- 1
Procedure foo (int x, int y)
  int temp
  temp <- x
  x <- y
  i <- 0
  y <- temp
end
A[0] <- 0
A[1] <- 2
foo (i, A[i])
write i, A[0], A[1]
```

7. **[10 pt.]** Estas dos funciones tienen la misma semántica, pero una de ellas no es declarativa. ¿Cuál no es declarativa y por qué no lo es?

```
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)
```

8. **[10 pt.]** Lea los siguientes textos y explique con sus propias palabras la diferencia entre abstracción y encapsulación. Incluya en su explicación los conceptos de interfaz e implementación.

Abstraction is a process where you show only "relevant" data and "hide" unnecessary details of an object from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.

Stackoverflow

In object oriented programming languages, encapsulation is used to refer to one of two related but distinct notions, and sometimes to the combination thereof:

- *A language mechanism for restricting direct access to some of the object's components.*
- *A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.*

Wikipedia

9. **[15 pt.]** Diagrame los estados por los que pasa la pila de ejecución en el siguiente programa en java, enfocándose únicamente en el manejo de excepciones, sin 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. Explique si hay alguna relación entre los `throws` y el `catch`, y cómo se puede llegar a generar la excepción que captura el `catch` si no se encuentra explícitamente en el código.

```
import java.io.*;

public class test {

    public static void main(String[] args)
                            throws FileNotFoundException, IOException {
        copy(args[0],args[1]);
    }

    public static void copy(String to, String from)
                            throws FileNotFoundException, IOException {

        int chances = 3;
        boolean success = false;
        BufferedReader br = null;
        BufferedWriter bw = null;
        while (!success) {
            try {
                FileReader fr = new FileReader(from);
                FileWriter fw = new FileWriter(to);
                br = new BufferedReader(fr);
                bw = new BufferedWriter(fw);
```

```
                String line;

                line = br.readLine();
                while (line != null) {
                    System.out.println(line);
                    bw.write(line);
                    line = br.readLine();
                }
                success = true;
            }
            catch (SecurityException e) {
                if (chances == 0)
                    break;
                chances--;
                System.out.println("Security Exception");
                System.out.println("Update file permisssions.");
                byte [] bytes = new byte[3];
                System.in.read(bytes);
            }
            finally {
                try {
                    if (br != null)
                        br.close();
                    if (bw != null)
                        bw.close();
                }
                catch(IOException e) {}
            }
        }
    }
}
```