Paradigmas de la Programación – Recuperatorio Primer Parcial

22 de Junio de 2017

Apellido y Nombre: _____

1. **[15 pt.]** Diagrame los sucesivos estados por los que pasa la pila de ejecución al ejecutar el siguiente programa, teniendo en cuenta que el lenguaje tiene alcance estático.

```
Program A()
{   x, y, z: integer;

    procedure B()
    {   y: integer;
        y=0;
        x=z+1;
        z=y+2;
    }

    procedure C()
    {   z: integer;

        procedure D()
        {   x: integer;
            x = z + 1;
            y = x + 1;
            call B();
        }
        z = 5;
        call D();
    }
    x = 10;
    y = 11;
    z = 12;
    call C();
    print x, y, z;
}
```

2. **[15 pt.]** En el siguiente código en Java, explique qué sucede (cuáles son los cambios en los sucesivos estados de la máquina, a nivel de pila de ejecución) cuando se produce una excepción aritmética o una excepción de cualquier otro tipo.

```
class Clase {
   public static void main(String args[]){
     try{
        System.out.println("primera sentencia del bloque try");
        ***CODIGO QUE LANZA UNA EXCEPCION ARITMETICA O DE OTRO TIPO***;
        System.out.println(num);
     }
```

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

3. **[15 pt.]** En Javascript no hay clases pero sí objetos. Vea cómo se instancian objetos en el siguiente código de Javascript y explique cuál sería la función de la palabra clave `constructor` estableciendo un paralelismo con lenguajes como C++ o Java.

   Nota: Esta pregunta salió en el parcial correspondiente, así que requiero mucha más precisión en las respuestas para ser evaluadas positivamente.

```
function User (theName, theEmail) {
    this.name = theName;
    this.email = theEmail;
    this.currentScore = 0;
}

User.prototype = {
    constructor: User,
    saveScore:function (theScoreToAdd)  {
        this.quizScores.push(theScoreToAdd)
    },
    changeEmail:function (newEmail)  {
        this.email = newEmail;
        return "New Email Saved: " + this.email;
    }
}
```

4. **[15 pt.]** Elm es un lenguaje de programación que pretende sustituir Javascript. Es un lenguaje de programación declarativo, sin embargo pretende usarse en los mismos contextos que un lenguaje de programación de scripting tipo glue. La propuesta es circunscribir las partes no declarativas de los programas fuera de las componentes puramente declarativas, construyendo programas dentro de la llamada The Elm Architecture (TEA). Compare la funcionalidad de la TEA con la funcionalidad de las mónadas en otros lenguajes funcionales (como Haskell), explique las propiedades de las componentes declarativas y cómo estas propiedades producen limitaciones en los lenguajes de programación, y cómo estas limitaciones se pueden solventar mediante la TEA o las mónadas.

5. **[10 pt.]** En el siguiente código, ¿puedo asegurar si este lenguaje tiene pasaje de parámetros por *call-by-value*, *call-by-reference*, *call-by-value-resultx*, *call-by-need* o *call-by-name*? ¿Qué tipos de pasaje por parámetros voy a poder distinguir según el output?

```
    begin
    array a[1..10] of integer;
    integer n;
    procedure p(b: integer);
        begin
        print(b);
        n := n+1;
        print(b);
        end;
```

```
a[1] := 10;
a[2] := 20;
a[3] := 30;
a[4] := 40;
n := 1;
p(a[n+2]);
print(a);
end;
```

6. **[15 pt.]** Explique el *overhead* que producen las abstracciones lingüísticas propias de los lenguajes orientados a objetos. Explique qué diferencias hay entre lenguajes que por defecto resuelven estas abstracciones en tiempo de compilación (como C++) y los que las resuelven en tiempo de ejecución (como Smalltalk), en términos de flexibilidad, reemplazo en caliente, overhead en tiempo de ejecución y complejidad del compilador.

7. **[15 pt.]** Explique por qué es importante tener un compilador con optimizaciones, y cómo eso hace una diferencia entre lenguajes de programación. Explique en qué decisión de diseño impacta esa diferencia. Describa la optimización de recursión a la cola y explique en qué puede ayudar esa optimización.