Paradigmas de la Programación – Examen Final

5 de Julio de 2019

Apellido y Nombre: _____

1. **[10 pt.]** Diagrame los sucesivos estados por los que pasa la pila de ejecución del siguiente programa cando se ejecuta N(3), asumiendo que el lenguaje tiene alcance estático. ¿Qué se imprimirá si el lenguaje tiene alcance estático? ¿Y si tiene alcance dinámico?

```
procedure N(y : int)
begin
  var z : int = 2;

  procedure Q(v : int)
  begin
    print v;
  end

  procedure P()
  begin
    print z;
  end

  if true begin
    var z : int = 42;
    call P();
  end

  print y;
  print z;

  call Q(y);
end
```

2. **[10 pt.]** En el siguiente fragmento de código, qué se imprimirá si el lenguaje tiene pasaje de parámetros por referencia, pasaje de parámetros por valor, y pasaje de parámetros 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 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 qué pasa con los dos `catch`, y por qué.

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            int a[]= {1, 2, 3, 4};
            for (int i = 1; i <= 4; i++)
            {
                System.out.println ("a[" + i + "]=" + a[i] + "n");
            }
        }
        catch (Exception e)
        {
            System.out.println ("error = " + e);
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            System.out.println ("ArrayIndexOutOfBoundsException");
        }
    }
}
```

4. **[10 pt.]** Dada la siguiente base de conocimiento en Prolog, liste los subobjetivos que hay que satisfacer para verificar si el predicado `tio(Pedro,Juan).` es cierto o no.

```
hermano(Pedro, Julia).
hermana(Ana, Teresa).
hermano(Pedro, Esteban).
madre(Ana, Juan).
padre(Esteban, Juan).

tio(X,Y) :- hermano(X,Z), progenitor(Z,Y).
progenitor(X,Y) :- padre(X,Y).
progenitor(X,Y) :- madre(X,Y).
```

5. **[10 pt.]** En el siguiente programa en AWK, identifique por lo menos dos características propias de los lenguajes de scripting y justifíquelas con porciones del código.

```
if ((x=index($1,"@")) > 0) {
    username = substr($1,1,x-1);
    hostname = substr($1,x+1,length($1));
    printf("username = %s, hostname = %s\n", username, hostname);
}
```

6. **[5 pt.]** La siguiente expresión está mal tipada:

```
f (g, x) = g(x) > 3 + g
```

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?

7. **[5 pt.]** De estos dos fragmentos de códigos, cuál es un ejemplo de polimorfismo y cuál es un ejemplo de sobrecarga? Justifique su respuesta.

```
public class Animal{
   public void sound(){
      System.out.println("Animal is making a sound");
   }
}

class Horse extends Animal{
    @Override
    public void sound(){
        System.out.println("Neigh");
    }
}

public class Cat extends Animal{
    @Override
    public void sound(){
        System.out.println("Meow");
    }
}
```

```
class Examp
{
    void demo (int a)
    {
       System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
       System.out.println ("a and b: " + a + "," + b);
    }
    double demo(double a) {
       System.out.println("double a: " + a);
       return a*a;
    }
}
class MethodExamping
{
    public static void main (String args [])
    {
        Examp Obj = new Examp();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

8. **[15 pt.]** Explique por qué el siguiente programa es un ejemplo de programación defensiva, identifique las líneas de código que implementan programación defensiva (5 pt.). Explique qué vulnerabilidad del lenguaje se está protegiendo mediante estas acciones, y cómo otros lenguajes evitan este tipo de vulnerabilidad (5 pt.). Por último, escriba en pseudocódigo cómo se podría tratar este problema con excepciones (5 pt.).

```
static void CreateRandomPermutation(int numbers[], int nNumbers)
{
    assert(numbers != NULL);
    assert(nNumbers >= 0);

    for (int i=0; i<nNumbers; i++)
        numbers[i] = i;

    for (int src=1; src<nNumbers; src++)
    {
        const int dst = GetRandomNumberIn(0, src);
        SwapNumbers(numbers, src, dst);
    }
}
```

9. **[5 pt.]** Lea el siguiente texto y explique con sus propias palabras el nivel de visibilidad `package` en Java, y relaciónelo con los niveles de visibilidad `private` y `protected`.

> *Access level modifiers determine whether other classes can use a particular field or invoke a particular method. There are two levels of access control:*
>
> - *At the top level—public, or package-private (no explicit modifier).*
> - *At the member level—public, private, protected, or package-private (no explicit modifier).*
>
> *A class may be declared with the modifier public, in which case that class is visible to all classes everywhere. If a class has no modifier (the default, also known as package-private), it is visible only within its own package (packages are named groups of related classes).*
>
> *At the member level, you can also use the public modifier or no modifier (package-private) just as with top-level classes, and with the same meaning. For members, there are two additional access modifiers: private and protected. The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package (as with package-private) and, in addition, by a subclass of its class in another package.*

10. **[10 pt.]** De los siguientes fragmentos de código, ¿cuál es declarativo? En el que no es declarativo ¿Qué fenómenos podemos encontrar que no son declarativos?

```
procedure FACTORIAL is
   N: integer;
   T: integer:= 1;
begin
   put("Input ");get(N);
   for K in 2..N loop
      T:= T*K;
   end loop;
   put("Factorial ");put(N);
   put("is ");put(T);newline;
end FACTORIAL;
```

```
factorial(0,1):-
        !.

factorial(N1,T2):-
        N2 is N1-1,
        factorial(N2,T1),
        T2 is N1*T1.
```

Ejercicios para libres

1. **[10 pt.]** En el siguiente código en C++, hay un problema con la herencia del miembro "meow", que viene dada por el tipo de herencia por defecto que tiene C++. Explique por qué C++ tiene herencia no virtual, sin reescritura, por defecto, y cómo habría que hacer para que este programa funcionara.

```
class Felino {
public:
 void meow() = 0;
};

class Gato : public Felino {
public:
 void meow() { std::cout << "miau\n"; }
};

class Tigre : public Felino {
public:
 void meow() { std::cout << "ROARRRRRR\n"; }
};

class Ocelote : public Felino {
public:
 void meow() { std::cout << "roarrrrr\n"; }
};
```

2. **[10 pt.]** En el siguiente programa en Erlang, identifique los fragmentos de programa con semántica concurrente y explique cuál es su semántica y por qué es concurrente, es decir, qué semántica expresan que no está incluída en la expresividad de las máquinas de Turing. Identifique, en particular, cómo se expresa sincronización, exclusión mútua, generación de procesos y monitoreo de procesos.

   Note que alguna de las palabras del programa se puede usar para más de una de estas funciones.

```
max(N) ->
    Max = erlang:system_info(process_limit),
    io:format("Maximum allowed processes:~p~n" ,[Max]),

    statistics(runtime),
    statistics(wall_clock),

    L = for(1, N, fun() -> spawn(fun() -> wait() end) end),
    {_, Time1} = statistics(runtime),
    {_, Time2} = statistics(wall_clock), lists:foreach(fun(Pid) -> Pid ! die end, L

    U1 = Time1 * 1000 / N,
```

```
    U2 = Time2 * 1000 / N,
    io:format("Process spawn time=~p (~p) microseconds~n" , [U1, U2]).
    wait() ->

    receive
        die -> void
    end.

for(N, N, F) -> [F()];
for(I, N, F) -> [F()|for(I+1, N, F)].

start()->
    max(1000),
    max(100000).
```

3. **[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:
        ...
```