Paradigmas de la Programación – Segundo Parcial - Segunda Fecha

16 de Junio de 2022

Apellido y Nombre: _____

1	
2	
3	
4	

5	
6	
7	
8	

1. **[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.

```
set found ''
cat /etc/fstab | while read dev mnt rest
  if test "$mnt" = "/"
    set found $dev
  end
end
```

2. **[15 pt.]** Lea el siguiente texto:

> *If Await.Result() is called at any point before the Future has completed, the Future becomes blocking. If you instead use onComplete, onSuccess, onFailure, map, or flatMap (and some other methods), you are registering a callback function that will occur when the Future returns. Thus, the Future is non-blocking. Use non-blocking Futures with callbacks whenever possible.*

Este texto nos habla de una propiedad interesante de los futuros. ¿Qué efectos tiene esta propiedad con respecto a la eficiencia de los programas orientados a actores? ¿Con qué decisión de diseño de la orientación a actores la pueden relacionar, y por qué?

3. [**10 pt.**] Al ejecutar el fragmento de código de arriba en un contexto concurrente, con otros hilos corriendo el mismo código u otro código que modifique "*balance*", podemos tener problemas. ¿Cuál es el comportamiento del programa que puede producir efectos difíciles de prever? [**10 pt.**] ¿Cómo se previenen esos efectos con la orientación a actores?

```
1 bool withdraw(int withdrawal)
2 {
3     if (balance >= withdrawal)
4     {
5         balance -= withdrawal;
6         return true;
7     }
8     return false;
9 }
```

4. [**10 pt.**] A partir de la siguiente base de conocimiento, ¿qué subobjetivos se tienen que satisfacer para probar que es cierto que `alivia(aspirina,gripe).`?

```
enfermo_de(manuel,gripe).
tiene_sintoma(alicia,cansancio).
sintoma_de(fiebre,gripe).
sintoma_de(tos,gripe).
sintoma_de(cansancio,anemia).
elimina(vitaminas,cansancio).
elimina(aspirinas,fiebre).
elimina(jarabe,tos).
recetar_a(X,Y):-enfermo_de(Y,A),alivia(X,A).
alivia(X,Y):-elimina(X,A),sintoma_de(A,Y).

enfermo_de(X,Y):-tiene_sintoma(X,Z),sintoma_de(Z,Y).
```

5. [**15 pt.**] En el siguiente texto se explica el comportamiento de los streams en programación reactiva.

> *In computing, reactive programming is a declarative programming paradigm concerned with data streams and the propagation of change. With this paradigm, it's possible to express static (e.g., arrays) or dynamic (e.g., event emitters) data streams with ease, and also communicate that an inferred dependency within the associated execution model exists, which facilitates the automatic propagation of the changed data flow.*
> *For example, in an imperative programming setting,* `a := b + c` *would mean that a is being assigned the result of* `b + c` *in the instant the expression is evaluated, and later, the values of* `b` *and* `c` *can be changed with no effect on the value of* `a`*. On the other hand, in reactive programming, the value of* `a` *is automatically updated whenever the values of* `b` *or* `c` *change, without the program having to explicitly re-execute the statement* `a := b + c` *to determine the presently assigned value of* `a`*.*

Según esta definición, explique qué se imprimirá al ejecutar el siguiente programa si el operador `"="` se interpreta como el operador de asignación propio de la programación imperativa, o bien si se interpreta como un operador reactivo, que cambia el valor de la variable del lado izquierdo del operador no solamente cuando se hace la asignación explícitamente, sino también cuando las variables referenciadas en la parte derecha del operador cambian.

```
var b = 1
var c = 2
var a = b + c
b = 10
console.log(a)
```

6. [**10 pt.**] El siguiente código forma parte de un código de framework. ¿Cómo se llama la parte del código en la que se encuentra el string "`/user/:id`"?

```
<div id="app">
  <router-view></router-view>
</div>
...

<script>
...
const User = {
  template: '<div>User {{ $route.params.id }}</div>'
}

const router = new VueRouter({
  routes: [
    { path: '/user/:id', component: User }
  ]
})
...
</script>
```

7. [**5 pt.**] En el siguiente programa, identifique los mecanismos de programación defensiva, y [**10 pt.**] explique qué tipo de inseguridad se está tratando de cubrir con esos mecanismos. [**5 pt.**] ¿Cómo se escribiría este programa en programación ofensiva? ¿Cuál es el objetivo de seguridad de escribir un programa como este en su versión ofensiva?

```
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);
    }
}
```