Construcción Formal de Programas en Teoría de Tipos
Primer Parcial – Octubre de 2017

Problema 1. Demuestre la validez del siguiente razonamiento.
Si el general era leal, habría obedecido las órdenes, y si era inteligente las habría comprendido. O el general desobedeció las órdenes o no las comprendió. Luego, el general era desleal o no era inteligente.

Problema 2. Pruebe el `lema2` en Coq usando lógica clásica, teniendo en cuenta las siguientes definiciones:

```
Section Problema2.
Variable C : Set.
Variable P : C -> C -> Prop.
Lemma lema2 : (exists x y : C, P x y) \/ ~(exists x : C, P x x).
...

End Problema2.
```

Problema 3. Considere las declaraciones:

```
Variable U : Set.
Variable a : U.
Variables P Q R T : U -> Prop.
```

a) Complete la siguiente prueba en Coq utilizando únicamente una aplicación de la táctica `exact` con el término asociado.

```
Lemma Ej3_1 : (forall x : U, P x -> Q x) -> P a -> Q a.
Proof.
...

Lemma Ej3_2 : (forall x : U, P x -> Q x) -> (forall x : U, Q x -> R x) ->
                  forall x : U, P x -> R x.
Proof.
intros.
...
```

b) Demuestre el siguiente lema en un sólo renglón utilizando compositores (estructuradores) de tácticas:

```
Lemma L3_3: (forall x:U, Q x) \/ (forall y:U, T y) -> forall z:U, Q z \/ T z.
Proof.
...
```

Problema 4. Considere el constructor de tipos `ABnat` de árboles binarios no vacíos de `n` nodos, de números naturales:

```
Parameter ABnat : forall n : nat, Set.
```

a) Defina el tipo del operador `null` que permita representar un árbol binario de naturales vacío (sin nodos).
b) Defina el tipo del operador `add` que permita generar árboles binarios no vacíos de naturales. Este operador junto con `null` deberían permitir crear cualquier árbol de naturales de `n` nodos.
c) Construya utilizando los operadores `null` y `add` un árbol binario de naturales que tenga 3 nodos y sea de altura 3, cuyos elementos sean el 7, el 8 y el 9 (en cualquier orden).
d) Generalice las definiciones `ABnat`, `null` y `add` para trabajar con elementos de un tipo genérico (en vez de `nat`).

NOTA: en todo el problema, use el tipo nat predefinido de Coq con `O`, `S` y plus (`+`), como el cero, el sucesor y la suma de naturales, respectivamente.

Problema 5. Considere las siguientes declaraciones:

```
Section Ej5.

Variable Bool: Set.
Variable TRUE : Bool.
Variable FALSE : Bool.
Variable Not : Bool -> Bool.
Variable Imp : Bool -> Bool -> Bool.
Variable Xor : Bool -> Bool -> Bool.

Axiom Disc : ~ (FALSE = TRUE).
Axiom BoolVal : forall b : Bool, b = TRUE \/ b = FALSE.
Axiom NotTrue : Not TRUE = FALSE.
Axiom NotFalse : Not FALSE = TRUE.
Axiom ImpFalse : forall b : Bool, Imp FALSE b = TRUE.
Axiom ImpTrue : forall b : Bool, Imp TRUE b = b.
Axiom XorTrue : forall b : Bool, Xor TRUE b = Not b.
Axiom XorFalse : forall b : Bool, Xor FALSE b = b.
```

Bajo el contexto previo, demuestre en Coq los siguientes lemas:

```
Lemma L51 : forall b: Bool, Xor b b = FALSE.

Lemma L52: forall b1 b2: Bool, Imp b1 b2 = FALSE -> b1 = TRUE /\ b2 = FALSE.
```