1. Deduce el tipo de la expresión:

```
[A:Prop; H:(A\/~A->A); H0:(~A)](H0 (H (or_intror A ~A H0))).
```

sabiendo que:

```
Inductive or [A:Prop; B:Prop] : Prop :=
      or_introl : A->(or A B) | or_intror : B->(or A B)
```

Resultado

```
(A:Prop)(A\/~A->A)->~~A
```

dado que `(or_intror A ~A):~A->A\/~A` y, siendo `H0:(~A)`, se tiene que `(or_intror A ~A H0):A\/~A`. Entonces el tipo de `(H (or_intror A ~A H0))` será `A` y, como `H0:A->False`, el tipo de la expresión `(H0 (H (or_intror A ~A H0)))` será `False`. Así pues, en virtud de la regla *lam* de tipado, se tiene que el tipo pedido será:

```
(A:Prop; H:(A\/~A->A); H0:(~A))False
```

Pero, como en `False` no figura `H0` se puede escribir:

```
(A:Prop; H:(A\/~A->A))(~A)->False
```

y como en `(~A)->False` no figura `H` se puede escribir:

```
(A:Prop)(A\/~A->A)->(~A)->False
```

2. Explica el siguiente comportamiento de coq:

```
 Fixpoint suma [n:nat] : nat->nat :=
    [m:nat]Cases n of
             O => m
           | (S p) => (S (suma p m))
           end.
suma is recursively defined


 Fixpoint suma [n:nat] : nat->nat :=
    [m:nat]Cases m of
             O => n
           | (S p) => (S (suma n p))
           end.
Error:
Recursive call applied to an illegal term
The recursive definition suma :=
[n,m:nat]Cases m of
           O => n
         | (S p) => (S (suma n p))
         end is not well-formed
```

[1] Calificación: 1:(2pt);2:(2pt);3:(3pt);4:(3pt)

Resultado Coq tiene un mecanismo para prevenir la construcción de programas recursivos, definidos sobre tipos inductivos, que no terminen. Para ello, se basa en el orden estructural de los elementos del tipo inductivo y comprueba que las llamadas al programa, dentro del cuerpo del mismo, se hacen siempre sobre valores estructuralmente menores que el que se pasa al programa.

En este caso, tratamos de definir la función `suma` sobre el tipo inductivo `nat` (cuyo orden es `O < (S O) ... n < (S n)`), como función de un argumento `n` y que nos devuelve una función de `nat->nat`.

La primera implementación es correcta porque el cálculo `(suma (S n))` se hace llamando a `(suma n)` (con un (primer) argumento menor).

La segunda no pasa el test de terminación porque el cálculo `(suma (S n))` se haría llamando a `(suma (S n))` otra vez (con un (primer) argumento igual).

3. Consideremos el tipo:

```
Inductive tree_nat:Set :=
  hoja:nat->tree_nat
 |cons:tree_nat -> tree_nat -> tree_nat.
```

Sabiendo que:

```
tree_nat_rec
     : (P:(tree_nat->Set))
        ((n:nat)(P (hoja n)))
        ->((t:tree_nat)(P t)->(t0:tree_nat)(P t0)->(P (cons t t0)))
        ->(t:tree_nat)(P t)
```

define, utilizando este esquema recursivo, una función `sumar_tree` que sume todas las hojas de un `t` de tipo `tree_nat`.

Resultado Como el resultado `(sumar_tree t)` ha de ser de tipo `nat`, sabemos que

```
sumar_tree:tree_nat -> nat
```

Por lo tanto, como el resultado de aplicar `tree_nat_rec` a sus tres argumentos ha de ser de tipo `(t:tree_nat)(P t)`, deducimos ya, que `P` (que es de tipo `tree_nat->Set`) ha de ser la función constante que a cualquier árbol de `tree_nat` le hace corresponder `nat` (que, efectivamente, es de tipo `Set`):

```
P = [_:tree_nat]nat
```

Tenemos así el primer parámetro para `tree_nat_rec`. Ahora, como segundo ingrediente hemos de tomar algo de tipo `(n:nat)(P (hoja n))`, es decir, de tipo `(n:nat)nat`, o lo que es lo mismo (dado que `n` no figura en `nat`), de tipo `nat -> nat`. ¿Qué elemento debemos de tomar? Dado que estamos en el caso en el que el árbol es una hoja, nuestra función, en este caso, debe devolver el mismo número que figura en esa hoja, por lo tanto el segundo ingrediente será:

```
[n:nat]n
```

Ahora debemos tomar un elemento que tenga de tipo:

```
(t:tree_nat)(P t)->(t0:tree_nat)(P t0)->(P (cons t t0))
```

es decir, con nuestro `P` concreto:

```
(t:tree_nat)nat->(t0:tree_nat)nat-> nat
```

que, simplificado (productos no dependientes) queda:

```
tree_nat -> nat-> tree_nat -> nat-> nat
```

De nuevo, ¿qué elemento debemos de tomar? Para responder a esta pregunta hemos de pensar que lo que tenemos que construir aquí es una función que a un `t:tree_nat`, a un `st:nat`, a un `t0:tree_nat` y a un `st0:nat` le hace corresponder un número de tal forma que si `st` es la suma de las hojas de `t` y `st0` es la suma de las hojas de `t0`, entonces ese número debe ser la suma de las hojas del árbol `(cons t t0)`. Así pues la función será:

```
[t:tree_nat][st:nat][t0:tree_nat][st0:nat](plus st st0)
```

Por lo tanto, la respuesta a la pregunta es:

```
Definition sumar_tree :=
(tree_nat_rec [_:tree_nat]nat
[x:nat]x
[t:tree_nat][st:nat][t0:tree_nat][st0:nat](plus st st0)).
```

Por ejemplo:

```
Require Arith.
Definition tree_nat1:=(cons (hoja (8)) (hoja (9))).
Definition tree_nat2:=(cons (hoja (1)) (hoja (0))).
Definition tree_nat3:=(cons tree_nat1 tree_nat2).
Coq <  Eval Compute in (sumar_tree (hoja (2))).
     = (2)
     : ([_:tree_nat]nat (hoja (2)))

Coq < Eval Compute in (sumar_tree tree_nat1).
     = (17)
     : ([_:tree_nat]nat tree_nat1)

Coq < Eval Compute in (sumar_tree tree_nat2).
     = (1)
     : ([_:tree_nat]nat tree_nat2)

Coq < Eval Compute in (sumar_tree tree_nat3).
     = (18)
     : ([_:tree_nat]nat tree_nat3)
```

Téngase en cuenta que:

```
Eval Simpl in ([_:tree_nat]nat tree_nat1).
     = nat
     : Set
```

4. Explica el significado de las tácticas `Elim` y `Case`. Ilústralo con algún ejemplo.

1. Calcula la función generatriz de la sucesión $a_n,\ n \in \mathrm{N}$, donde a_n es el número de formas de obtener la suma n cuando se lanzan nueve dados diferentes. Calcula a_{25}.

   Resultado Se trata de calcular el coeficiente del término de grado 25 del polinomio:

   $$f(x) = (x + x^2 + x^3 + x^4 + x^5 + x^6)^9$$

   que es lo mismo que:

   $$x^9 \left(\frac{1-x^6}{1-x}\right)^9$$

   Así pues, $a_{25} = \text{coef de } x^{16} \text{ en } \left(\frac{1-x^6}{1-x}\right)^9$.

   $\left(\frac{1-x^6}{1-x}\right)^9 = (1-x^6)^9 \left(\frac{1}{1-x}\right)^9 = \left(1 - \binom{9}{1}x^6 + \binom{9}{2}x^{12} - \cdots\right)\left(\sum_0^\infty \binom{8+i}{i}x^i\right)$

   Por lo tanto, $a_{25} = 1\binom{24}{16} - \binom{9}{1}\binom{18}{10} + \binom{9}{2}\binom{12}{4} = \frac{24!}{16!8!} - 9\frac{18!}{10!8!} + 36\frac{12!}{4!8!} = 359469$

2. Halla la solución de la relación de recurrencia

   $$a_n - 2a_{n-1} + a_{n-2} = 2^{n-2},\ \ n \geq 2$$

   con las condiciones iniciales $a_0 = 2$ y $a_1 = 1$.

   Resultado

   Formamos la expresión:

   $$a_n x^n - 2a_{n-1}x^n + a_{n-2}x^n = 2^{n-2}x^n,\ \ n \geq 2$$

   y, sumando:

   $$\sum_{n=2}^{\infty} a_n x^n - 2\sum_{n=2}^{\infty} a_{n-1}x^n + \sum_{n=2}^{\infty} a_{n-2}x^n = \sum_{n=2}^{\infty} 2^{n-2}x^n,\ \ n \geq 2$$

   es decir:

   $$\sum_{n=2}^{\infty} a_n x^n - 2x\sum_{n=2}^{\infty} a_{n-1}x^{n-1} + x^2\sum_{n=2}^{\infty} a_{n-2}x^{n-2} = x^2\sum_{n=2}^{\infty} 2^{n-2}x^{n-2},\ \ n \geq 2$$

   Sea $f(x) = \sum_{n=0}^{\infty} a_n x^n$, entonces se tiene:

   $$(f(x) - 2 - x) - 2x(f(x) - 2) + x^2 f(x) = x^2\frac{1}{1-2x}$$

   de donde:

   $$(1 - 2x + x^2)f(x) = \frac{x^2}{1-2x} - 3x + 2$$

   y se obtiene:

   $$f(x) = \frac{x^2}{(1-2x)(1-x)^2} + \frac{2-3x}{(1-x)^2}$$

   Ahora, con esta fracción generatriz podemos seguir de dos formas:

   Opción a)

[2] Calificación: 1:(5pt);2:(5pt)

Descomponemos en fracciones simples el primer sumando:

$$\frac{x^2}{(1-2x)(1-x)^2} = \frac{a}{1-2x} + \frac{b}{1-x} + \frac{c}{(1-x)^2} = \frac{a(1-x)^2 + b(1-2x)(1-x) + c(1-2x)}{(1-2x)(1-x)^2},$$

de donde, igualando $x^2 = a(1-x)^2 + b(1-2x)(1-x) + c(1-2x)$ se tiene:

$$\begin{aligned} x = 1 &\ ; \ c = -1 \\ x = 0 &\ ; \ a + b + c = 0 \text{ y } b = 0 \\ x = 1/2 &\ ; \ a = 1 \end{aligned}$$

obteniéndose:

$$f(x) = \frac{1}{1-2x} + \frac{-1}{(1-x)^2} + \frac{2-3x}{(1-x)^2} = \frac{1}{1-2x} + \frac{1}{(1-x)^2} + \frac{-3x}{(1-x)^2}$$

por lo tanto

$$\begin{aligned} a_n &= 2^n + \binom{n+1}{n} - 3\binom{n}{n-1};\ n \geq 1 \\ a_0 &= 1 + 1 - 0 = 2 \\ a_n &= 2^n + (n+1) - 3n = 2^n - 2n + 1;\ n \geq 1 \end{aligned}$$

Comprobemos que este resultado es correcto. Claramente es cierto para $n = 0$ y $n = 1$. También para $n = 2$, pues $a_2 = 4 - 4 + 1 = 1$ y $2.1 - 2 + 2^{2-2} = 1$.

Supongamos ahora, por inducción, que es cierto para cualquier $k < n$. Entonces,

$$a_n = 2a_{n-1} - a_{n-2} + 2^{n-2} = 2(2^{n-1} - 2(n-1) + 1) - (2^{n-2} - 2(n-2) + 1) + 2^{n-2} = 2^n - 2n + 1$$

Opción b)

$$f(x) = \frac{x^2 + (2-3x)(1-2x)}{(1-2x)(1-x)^2} = \frac{2 - 7x + 7x^2}{(1-2x)(1-x)^2} = \frac{a}{1-2x} + \frac{b}{1-x} + \frac{c}{(1-x)^2},$$

de donde, igualando $2 - 7x + 7x^2 = a(1-x)^2 + b(1-2x)(1-x) + c(1-2x)$ se tiene:

$$\begin{aligned} x = 1 &\ ; \ c = -2 \\ x = 1/2 &\ ; \ a = 1 \\ x = 0 &\ ; \ 2 = a + b + c \text{ y } b = 3 \end{aligned}$$

obteniendose:

$$f(x) = \frac{1}{1-2x} + \frac{3}{1-x} - \frac{2}{(1-x)^2}$$

por lo tanto

$$a_n = 2^n + 3 - 2\binom{n+1}{n} = 2^n - 2(n+1) + 3 = 2^n - 2n + 1$$