1. Deduce el tipo de la expresión:

   ```
   [A,B:Prop; H:(A/\B)]Cases H of (conj H0 _) => H0 end
   ```

   sabiendo que:

   ```
   Inductive and [A : Prop; B : Prop]  : Prop :=  conj : A->B->A/\B
   ```

   Resultado Se trata de un programa que toma `A,B:Prop` y nos devuelve el programa que a un `H:A/\B` le hace corresponder la parte de `A` que compone `H`. Es decir, `H` ha de ser de la forma `(conj H0 H1)` con `H0:A` y `H1:B`. Por lo tanto el tipo pedido será

   ```
   (A,B:Prop)((H:A/\B)A)
   ```

   que, dado que `A` no depende de `H`, puede escribirse de forma más sencilla como:

   ```
   (A,B:Prop)((A/\B)->A)
   ```

   o, eliminando paréntesis innecesarios:

   ```
   (A,B:Prop)A/\B->A
   ```

2. Dado el tipo

   ```
   Coq < Inductive lista:nat -> Set :=
   Coq < lista_1: (lista O)
   Coq < |lista_c:(n:nat)(A:Set)A->(lista n)->(lista (S n)).
   ```

   - si `x:nat` y `t:bool`, deduce el tipo de

     ```
     (lista_c (1) nat x (lista_c O bool t (lista_1))).
     ```

     Resultado Se trata de la utilización del constructor `lista_c` para construir un valor del tipo `(lista (S n))` para algún `n`. Para saber cuál, basta ver el primer argumento que se le pasa al constructor que es `(1)` de modo que, si el resto de los argumentos tienen los tipos correctos, entonces el tipo resultante será `(lista (2))`. ¿Son los otros argumentos de los tipos adecuados? En efecto, `nat:Set`, `(1):nat` y `(lista_c O bool t (lista_1))` es de tipo `(lista O)` como es fácil ver.

   - ¿qué clase de objetos representa este tipo?
     Resultado Un `l:(lista n)` representa una lista heterogénea de longitud `n`.

   - simplifica la expresión siguiente y calcula su tipo:

     ```
     ([n:nat; _:(lista n)]nat (2) (lista_c (1) nat x (lista_c O bool t
     (lista_1))).
     ```

     Resultado Se trata de la aplicación del programa `[n:nat; _:(lista n)]nat` a dos argumentos del tipo esperado. Este programa, como se ve, hace corresponder lo mismo (`nat`) a cualesquiera argumentos. Así pues, toda esta expresión se reduce a `nat` que es de tipo `Set`.

   - definiendo:

[1] Calificación: 1:(2pt);2:(5pt);3:(3pt)

```
Fixpoint G [n:nat; l:(lista n)] : nat :=
    Cases l of
      lista_1 => (1)
    | (lista_c n0 A _ l0) => (S (G  n0 l0))
end.
```

calcula el resultado de las dos expresiones siguientes:

```
 (G (5) (lista_c (1) nat x (lista_c 0 bool t (lista_1)))).
 (G (2) (lista_c (1) nat x (lista_c 0 bool t (lista_1)))).
```

Resultado En la primera, el primer argumento no coincide con la longitud de la lista y por lo tanto dará un error.

En la segunda, el resultado será la longitud de la lista más uno, es decir `(3)`.

- teniendo en cuenta que el tipo de `lista_rec` es

```
(P:((n:nat)(lista n)->Set))
(P (0) lista_1)
 ->((n:nat; A:Set; a:A; l:(lista n))
(P n l)->(P (S n) (lista_c n A a l)))
        ->(n:nat; l:(lista n))(P n l)
```

reescribe la función `G` usando este esquema recursivo.

Resultado El primer argumento es `P`, y como el resultado de la función que intentamos obtener debe ser `nat` y del tipo `(n:nat; l:(lista n))(P n l)`, es claro que `P` debe ser:

```
[n:nat; _:(lista n)]nat
```

El siguiente argumento debe ser un término de tipo `(P (0) lista_1)` es decir, de tipo `nat`. Y ¿qué valor debemos elegir? pues el valor que debe devolver nuestro programa en el caso de que se aplique a `lista_1`. Pues bien, para que este valor coincida con el que nos daría `G`, deberá ser `(1)`.

Análogamente, se puede ver que el tercer argumento es:

```
[n:nat;A:Set;a:A;l:(lista n)][x:nat](S x)).
```

de modo que:

```
Definition Gr:=(lista_rec
[n:nat; _:(lista n)]nat
(1)
[n:nat;A:Set;a:A;l:(lista n)][x:nat](S x)).
```

Auque esto no forma parte de la respuesta que se espera en el examen, lo que sigue sería una posible demostración de que `G` y `Gr` funcionan exactamente igual:

```
Goal (n:nat; l:(lista n))(G n l)=(Gr n l).
Proof.
Intros.
Elim l.
Simpl.
Trivial.

Intros.
Simpl.
Rewrite H.
Trivial.
Qed.
```

3. Explica el fundamento del concepto de "táctica"e ilústralo con algunos ejemplos

1. Sea a_n el número de formas distintas de distribuir n bolas idénticas en m cajas numeradas, siendo m un número natural fijo, de forma que cada caja contenga una o dos bolas. Calcula la función generatriz de la sucesión $\{a_n\}$ y los valores de a_n, $n \geq 0$.

Resultado

$f(x) = (x + x^2)^m = \sum_{k=0}^{m} \binom{m}{k} x^k (x^2)^{m-k} = \sum_{k=0}^{m} \binom{m}{k} x^{2m-k}$ $m \in \mathbb{N}$

a_n es el coeficiente de grado n de $f(x)$ y valdrá, $\binom{m}{2m-n} = \binom{m}{n-m}$ si $m \leq n \leq 2m$ ó 0 en otro caso.

2. Sea b_n el número de listas de longitud n formadas con los símbolos $\{0, 1, 2, 3\}$ que tienen un número impar de ceros.

   a. demuestra que $b_{n+1} = 2b_n + 4^n$, $n \geq 1$.

   b. utilizando funciones generatrices calcula el valor de b_n.

Resultado

a) Sea $\alpha \equiv \alpha_1\ \alpha_2\ \ldots \alpha_n\ \alpha_{n+1}$ una lista válida de longitud $n+1$

caso 1. α_{n+1} es 1, 2 o 3; entonces $\alpha_1\ \alpha_2\ \ldots \alpha_n$ es una lista válida de longitud n y viceversa. De este tipo hay $3b_n$.

caso 2. $\alpha_{n+1} = 0$; entonces $\alpha_1\ \alpha_2\ \ldots \alpha_n$ es una lista no válida de longitud n y viceversa. De este tipo hay $4^n - b_n$.

En total habrá $3b_n + (4^n - b_n) = 2b_n + 4^n$ con $n \geq 1$. Dado que $b_0 = 0$ y $b_1 = 1$ se puede ampliar la relación para $n \geq 0$.

b) $b_{n+1} = 2b_n + 4^n$ y $b_{n+1}x^{n+1} = 2b_n x^{n+1} + 4^n x^{n+1}$ con $n \geq 1$.
Sumando todas las igualdades para $n \geq 1$

$$\sum_{n=1}^{\infty} b_{n+1}x^{n+1} = \sum_{n=1}^{\infty} 2b_n x^{n+1} + \sum_{n=1}^{\infty} 4^n x^{n+1}$$

o sea

$$\sum_{n=1}^{\infty} b_{n+1}x^{n+1} = 2x \sum_{n=1}^{\infty} b_n x^n + x \sum_{n=1}^{\infty} 4^n x^n$$

Sea $f(x) = \sum_{n=0}^{\infty} b_n x^n$ con $b_0 = 0$ y $b_1 = 1$. Entonces se tiene

$$f(x) - x = 2xf(x) + x(\frac{1}{1-4x} - 1) = 2xf(x) + \frac{x}{1-4x} - x$$

por tanto

$$f(x) = \frac{x}{(1-2x)(1-4x)} = \frac{-1/2}{1-2x} + \frac{1/2}{1-4x}$$

b_n será el coeficiente de x^n de $f(x)$:

$$b_n = \frac{1}{2}(4^n - 2^n) \quad n \geq 0.$$

Se puede ahora comprobar el resultado razonando por inducción:

[2] Calificación: 1:(3.4pt);2:(6.6pt)

$b_0 = \frac{1}{2}(4^0 - 2^0) = 0$, $b_1 = \frac{1}{2}(4 - 2) = 1$ y, suponiendo que $b_n = \frac{1}{2}(4^n - 2^n)$ entonces:

$$b_{n+1} = 2b_n + 4^n = 4^n - 2^n + 4^n = 2.4^n - 2^n = \frac{1}{2}(44^n - 22^n) = \frac{1}{2}(4^{n+1} - 2^{n+1})$$