1 **Teo**: Principio del Buen Orden ≡ Principio de inducción.
(⇒) Supóngase el Principio del Buen Orden.
Sea p un predicado sobre **nat** tal que p.0 y (∀n:**nat**|: p.n ⇒ p(n+1)}. Se quiere demostrar que también vale (∀n:**nat**|: p.n).
Sea A = {n:**nat**| p.n}. Si $A^c \neq \emptyset$, el Principio del Buen Orden garantiza que A^c tiene un primer elemento a. Como 0∈A, porque p.0 vale, debe ser a>0. Como $a \in A^c$ y a es el primer elemento de este conjunto, a-1∈A, i.e., p(a-1) vale. Por tanto p(a-1+1) también vale, de modo que a∈A, lo que es una contradicción. Por lo tanto, $A^c = \emptyset$ y, así mismo, A = **nat**, o bien, (∀n:**nat**|: p.n).

(⇐) Sea X un subconjunto no vacío de **nat**. Supóngase que X no tiene un primer elemento. Para n∈**nat**, sea B_n el intervalo 0..n. Considérese el predicado p.n ≡ $B_n \subseteq X^c$. Nótese que $B_0 \subseteq X^c$, porque B_0={0} y 0∉X porque, de lo contrario, 0 sería un primer elemento para X. Ahora, si $B_n \subseteq X^c$, entonces $B_{n+1} = B_n \cup \{n+1\} \subseteq X^c$, ya que $n+1 \in X^c$ porque, en otro caso, n+1∈X y n+1≤x para todos los elementos de X, de modo que x+1 sería un primer elemento. Es decir, p.n ⇒ p(n+1). Por el Principio de Inducción Simple, vale (∀n:**nat**|: $B_n \subseteq X^c$). Entonces: **nat** = (∪n:**nat**|: B_n) ⊆ (∪n:**nat**|: X^c) = X^c. Así, X=∅, lo cual es una contradicción. Por tanto, la suposición de que X no tiene primer elemento es falsa.

2

a (+k| 0≤k≤n : k^2) = n(n+1)(2n+1)/6
Dem:
Inducción sobre **nat.**
Predicado de inducción: p.n ≡ (+k| 0≤k≤n : k^2) = n(n+1)(2n+1)/6, n≥0.

Caso base: p.0
(+k| 0≤k≤0 : k^2)
= ⟨0≤k≤0 ≡ k=0; 1 pto⟩
0
= ⟨aritmética⟩
0(0+1)(2*0+1)/6

Caso Inductivo: p(n+1)
HI: p.n, n≥0
(+k| 0≤k≤n+1 : k^2)
= ⟨Partir rango a la derecha⟩
(+k| 0≤k≤n : k^2)+$(n+1)^2$
= ⟨HI⟩
n(n+1)(2n+1)/6+$(n+1)^2$
= ⟨aritmética⟩
(n+1)((n+1)+1)(2(n+1)+1)/6

e (+k| 0≤k≤n : 3^k) = (3^{n+1} – 1)/2
Dem:
Inducción sobre **nat.**
Predicado de inducción: p.n ≡ (+k| 0≤k≤n : 3^k) = (3^{n+1} – 1)/2, n≥0.

Caso base: p.0
(+k| 0≤k≤0 : 3^k)

$=$ ⟨$0 \le k \le 0 \equiv k=0$; 1 pto⟩
3^0
$=$ ⟨aritmética⟩
1
$=$ ⟨aritmética⟩
$(3^1 - 1)/2$

Caso Inductivo: p(n+1)
HI: p.n, $n \ge 0$
$(+k|\ 0 \le k \le n+1 : 3^k)$
$=$ ⟨Partir rango a la derecha⟩
$(+k|\ 0 \le k \le n : 3^k) + 3^{n+1}$
$=$ ⟨HI⟩
$(3^{n+1} - 1)/2 + 3^{n+1}$
$=$ ⟨aritmética⟩
$(3^{n+2} - 1)/2$

f $n^2 + n$ es par.
Dem:
Inducción sobre **nat.**
Predicado de inducción: $p.n \equiv n^2 + n \equiv_2 0$, $n \ge 0$.

Caso base: p.0
$0^2 + 0$
$\equiv_2$ ⟨aritmética⟩
0

Caso Inductivo: p(n+1)
HI: p.n, $n \ge 0$
$(n+1)^2 + (n + 1)$
$=$ ⟨aritmética⟩
$(n^2 + n) + 2(n + 1)$
$\equiv_2$ ⟨HI⟩
$0 + 2(n + 1)$
$\equiv_2$ ⟨$2*k \equiv_2 0$⟩
$0.$

Ejercicios 8.2

1 Pruebe que el Principio de Inducción Simple es equivalente al Principio de Inducción Fuerte.
($\Leftarrow$) Trivial.
($\Rightarrow$) Supóngase el Principio de Inducción Simple. Se mostrará que el Principio de Inducción Fuerte debe valer. La prueba se hace usando como dominio de inducción (**nat**,$\le$), y resulta muy fácil de generalizar para el caso de un segmento final de **int**.
Sea p:**nat** $\to$ **bool** un predicado sobre **nat** que cumple que

$(\forall k:\mathbf{nat}|\ (\forall j|: 0 \le j < k : p.j): p.k)$ [*]

Sea q el predicado q:**nat** $\to$ **bool**, tal que

$q.k \equiv (\forall j|: 0 \le j < k : p.j)$ [**]

Nótese que [*] también se puede expresar equivalentemente como

$(\forall k:\mathbf{nat}|: q.k \Rightarrow p.k)$ [*a]

Obsérvese que q.0 debe valer, por rango vacío. Entonces también debe valer p.0, porque q.0 $\Rightarrow$ p.0. Ahora, si q.k vale,para k$\geq$0, entonces

```
       q(k+1)
 =           ⟨Def q⟩
       (∀j|: 0≤j<k+1 : p.j)
 =           ⟨Partir rango a la derecha⟩
       (∀j|: 0≤j<k : p.j) ∧ p.k
 =           ⟨[*a]⟩
       q.k ∧ p.k
 ⇐           ⟨q.k ⇒ p.k⟩
       q.k
```

Es decir, se cumple q.k $\Rightarrow$ q(k+1), k$\geq$0. Por el Principio de Inducción Simple, debe valer ($\forall$n|: q.n). Y, como q.n $\Rightarrow$ p.n para n$\geq$0, también ($\forall$n|: p.n).

2 Sea f.n el número de bits en 1 en la notación binaria estándar de n. Formalmente, se puede definir f: nat $\rightarrow$ nat por los siguientes axiomas:

$$f.0 = 0$$
$$f(2*n) = f.n \quad , 0<k<2^n$$
$$f(2*n+1) = 1+f.n, \quad 0<k<2^n$$

a Muestre que $f(2^n) = 1$, para 0$\leq$n.

Dem: Inducción sobre 0$\leq$n.

Predicado de inducción: p.n $\equiv$ $f(2^n) = 1$, para 0$\leq$n.

Caso base: p.0

```
       f(2^0)
 =
       f.1
 =
       f(2*0+1)
 =           ⟨Def f⟩
       1+f(0)
 =           ⟨Def f⟩
       1+0
 =
       1.
```

Caso Inductivo: p(n+1)

HI: p.n

```
       f(2^(n+1))
 =
       f(2*2^n)
 =           ⟨Def f⟩
       f(2^n)
 =           ⟨HI⟩
       1.
```

b Muestre que $f(2^n-1) = 1$, para 0<n.

Dem: Inducción fuerte sobre 0$\leq$n.

Predicado de inducción: q.n $\equiv$ $f(2^n-1) = 1$, para 0<n.

Caso base: q.1

```
       f(2^1-1)
 =
       f.1
 =           ⟨1 = 2*0+1⟩
```

$f(2*0+1)$
$= \quad \langle$Def f$\rangle$
$f.0+1$
$= \quad \langle$Def f$\rangle$
$0+1$
$=$
1

Caso Inductivo: q(n+1)
HI: q.0,…,q.n
$f(2^{n+1}-1)$
$= \quad \langle 2^{n+1}-1 = 2*2^n-1 \rangle$
$f(2*2^n - 1)$
$= \quad \langle 2*2^n-1 = 2*(2^n-1)+1 \rangle$
$f(2*(2^n-1)+1)$
$= \quad \langle$Def f$\rangle$
$f(2^n-1)+1$
$= \quad \langle$HI: $f(2^n-1)=n \rangle$
$n+1$

3 **a** Si $a_0=1$, $a_1=2$ y $a_{n+2}=2a_{n+1}+a_n$, entonces $a_n \le (5/2)^n$

Dem:
Inducción fuerte sobre **nat.**
Predicado de inducción: p.n ≡ $a_n \le (5/2)^n$, $n \ge 0$.

Caso base: p.0
a_0
$= \quad \langle$Def$\rangle$
1
$= \quad \langle$aritmética$\rangle$
$(5/2)^0$

Caso base: p.1
a_1
$= \quad \langle$Def$\rangle$
2
$\le \quad \langle$aritmética$\rangle$
2.5
$= \quad \langle$aritmética$\rangle$
$(5/2)^1$

Caso Inductivo: p(n+2)
HI: p.n, p(n+1), $n \ge 0$
a_{n+2}
$= \quad \langle$Def$\rangle$
$2a_{n+1}+a_n$
$\le \quad \langle$HI$\rangle$
$2(5/2)^{n+1} + (5/2)^n$
$= \quad \langle$aritmética$\rangle$
$5(5/2)^n + (5/2)^n$
$= \quad \langle$Factorización$\rangle$
$6(5/2)^n$
$\le \quad \langle 6 \le 25/4 \rangle$
$(5/2)^2(5/2)^n$

= ⟨aritmética⟩
$(5/2)^{n+2}$

b Si F_n es el n-ésimo número de Fibonacci, $F_n<2^n$
Dem:
Inducción fuerte sobre **nat.**
Predicado de inducción: p.n ≡ $F_n<2^n$, n≥0.

Caso base: p.0
F_0
= ⟨Def⟩
0
< ⟨aritmética⟩
1
= ⟨aritmética⟩
2^0

Caso base: p.1
F_1
= ⟨Def⟩
1
< ⟨aritmética⟩
2
= ⟨aritmética⟩
2^1

Caso Inductivo: p(n+2)
HI: p.n, p(n+1), n≥0
F_{n+2}
= ⟨Def⟩
$F_{n+1}+F_n$
< ⟨HI⟩
$2^{n+1} + 2^n$
= ⟨aritmética⟩
$3*2^n$
< ⟨3<4⟩
$4*2^n$
= ⟨aritmética⟩
2^{n+2}

c Si $a_1=5$, $a_2=13$ y $a_{n+2}=5a_{n+1}-6a_n$, n≥1 entonces $a_n=2^n+3^n$
Dem:
Inducción fuerte sobre **nat.**
Predicado de inducción: p.n ≡ $a_n=2^n+3^n$, n≥1.

Caso base: p.1
a_1
= ⟨Def⟩
5
= ⟨aritmética⟩
2+3
= ⟨aritmética⟩
2^1+3^1

```
 Caso base: p.2
    a2
 =  〈Def〉
    13
 =  〈aritmética〉
    4+9
 =  〈aritmética〉
    2^2+3^2
```

$$
\begin{aligned}
& a_2 \\
= \; & \langle \text{Def} \rangle \\
& 13 \\
= \; & \langle \text{aritmética} \rangle \\
& 4+9 \\
= \; & \langle \text{aritmética} \rangle \\
& 2^2+3^2
\end{aligned}
$$

Caso Inductivo: p(n+2)

HI: p.n, p(n+1), $n\geq 1$

$$
\begin{aligned}
& a_{n+2} \\
= \; & \langle \text{Def} \rangle \\
& 5a_{n+1}-6a_n \\
= \; & \langle \text{HI} \rangle \\
& 5*(2^{n+1}+3^{n+1}) - 6*(2^n+3^n) \\
= \; & \langle \text{aritmética} \rangle \\
& 10*2^n + 15*3^n - 6*2^n - 6*3^n \\
= \; & \langle \text{aritmética} \rangle \\
& 4*2^n + 9*3^n \\
= \; & \langle \text{aritmética} \rangle \\
& 2^{n+2}+3^{n+2}
\end{aligned}
$$

Ejercicios 8.3

1 Considere los siguientes axiomas para definir `mcd:` **`nat`** × **`nat`** → **`nat`**, una función que implementa el método de calcular el máximo común divisor con restas:

```
mcd(x,y) =
      if   x=y    then x
      elsf x<y    then mcd(y,x)
      else mcd(x-y,y)
      fi
```

Explique por qué es ésta una buena definición.

Los cálculos de `mcd` deben terminar si se considera el orden lexicográfico en **`nat`**×**`nat`**. Se muestra que, para cualquier par de datos `x,y`, `mcd(x,y)` se define directamente (si los datos son iguales) o, eventualmente, como igual a un `mcd(x',y')`, con $(x,y) >_{lex} (x',y')$. Como no puede haber cadenas descendentes infinitas, el cálculo debe terminar, i.e., no recurrir. La corrección de lo que se calcula estriba en el hecho de que se mantiene invariante el `mcd` de los datos iniciales, por propiedades aritméticas conocidas.

2 Considere la función `f:` **`nat`** → **`nat`** definida como:

```
f.n   = 1          , 0≤n≤1
      = f(n/2)     , n>1, par.n,
      = f(3*n+1)   , n>1, impar.n.
```

Utilice los axiomas para calcular `f` en diferentes argumentos, v.gr., 32, 5, 7.

```
f.32 = f.16 = f.8 = f.4 = f.2 = f.1 = 1
f.5 = f.16 = … = 1
f.7 = f.22 = f.11 = f.40 = f.20 = f.10 = f.5 = … = 1.
```

1 Enriquezca el TAD `Lista[nat]` con axiomas que definan las siguientes funciones

a `suma: Lista → nat`

donde `suma.l` es la suma de los elementos de la lista `l`.

```
suma.vac = 0
suma(ins(l,x)) = suma.l + x
```

b `mcd: Lista → nat`

donde `mcd.l` es el máximo común divisor de los elementos de la lista `l`.

```
mcd.vac = 0
mcd(ins(l,x)) =
      if   tam.l = 0 then x
      elsf tam.l > 0 then mcd(ins(resto.l,mcd2(prim.l,x))

mcd2(x,y) =
      if   y = 0  then x
      else mcd2(y, x mod y)
      fi
```

La función auxiliar `mcd2` corresponde al Algoritmo de Euclides por divisiones. Su terminación se discutió en el Ejercicio 7.3.3.

2 Enriquezca el TAD `Arbin[X]` con axiomas que definan las siguientes funciones

a `completo: Arbin → bool`

donde `completo.a` si el árbol a es completo.

```
completo.arvac = true
completo(cons(a,x,b)) = alt.a = alt.b ∧ completo.a ∧ completo.b
```

b `preorden: Arbin → Lista`

donde `preorden.a` es la lista que corresponde a la secuencia en preorden de los elementos del árbol a.

```
preorden.arvac = vac
preorden(cons(a,x,b)) = concat(concat(ins(vac,x),preorden.a),preorden.b)
```

3 Considere las definiciones recursivas sobre **nat**, para detectar la paridad o no paridad de un número,

`par, impar: nat → bool`

con los axiomas:

```
[p1] par.0 = true
[p2] par(n+1) = impar.n
[i1] impar.0 = false
[i2] impar(n+1) = par.n
```

Es decir, ¿Está el cálculo de `par.n` e `impar.n` bien definido?

Para un `n`∈**nat**, considérese la cadena de los argumentos de llamadas a las funciones `par` e `impar` que se suceden al llamar la función `par.n`. Por ejemplo, para `par.4` se puede calcular así:

`par.4 = impar.3 = par.2 = impar.1 = par.0`

La cadena correspondiente es ⟨4,3,2,1,0⟩. En general, siempre se da el caso de que la llamada a `par.n` tiene una cadena de llamadas de la forma ⟨n,n-1,…,1,0⟩.

En primer lugar, esto permite afirmar que los cálculos terminan, porque las cadenas son estrictamente descedientes y (**nat**,<) es un buen orden.

Por otro lado, si el número de elementos de la cadena es impar, `par.n ≡ par.0`, i.e., `par.n ≡ true`. Esto coincide con el concepto de paridad pretendido. Es decir, `par` calcula lo que se quiso definir.

Un argumento análogo se puede realizar para `impar`.

4 Considere los Algoritmos de Euclides como se definieron en 7.3.1. Muestre que estos procesos terminan siempre que los datos de entrada satisfagan la precondición.
cf. Ejercicio 7.3.3.