



En este ejercicio, se pretende establecer una relación biunívoca entre cada programa generado por nuestra gramática, y los números naturales N . El lenguaje que podemos generar (al que llamaremos SAL-*Simple Algorithmic Language*) posee las siguientes características:

- 1) Las variables son de la forma x_i , con $i \geq 1$.
- 2) Las únicas operaciones permitidas pueden tener 5 formas diferentes:
 - 2.1) $x_i := 0$
 - 2.2) $x_i := x_j$
 - 2.3) $x_i := x_j + 1$
 - 2.4) $x_i := x_j - 1$ (Operación resta, tal que $0 - 1 = 0$)
 - 2.5) **while** $x_i \neq 0$ **do** S **od** (S = secuencia de operaciones)
- 3) Un programa está formado por una secuencia finita de operaciones de los tipos anteriores. Con estas 5 operaciones se pueden crear todos los programas posibles. A cada operación le daremos un número, en base al siguiente criterio:

<i>Sentencia</i>	<i>Código</i>
$x_i := 0$	$5 * (i-1)$
$x_i := x_j$	$5 * \sigma_i^2(i-1, j-1) + 1$
$x_i := x_j + 1$	$5 * \sigma_i^2(i-1, j-1) + 2$
$x_i := x_j - 1$	$5 * \sigma_i^2(i-1, j-1) + 3$
while $x_i \neq 0$ do S od	$5 * \sigma_i^2(i-1, \text{cód}(S)) + 4$

A una secuencia de operaciones $S = \langle S_1, S_2, S_3, \dots, S_{n-1}, S_n \rangle$, le asignaremos un número que vendrá dado por: $\text{cód}(S) = \sigma_1^2(n-1, \sigma_1^2(S_n, \sigma_1^2(S_{n-1}, \sigma_1^2(\dots, \sigma_1^2(S_3, \sigma_1^2(S_2, S_1))\dots)))$. Un programa se considera una secuencia de operaciones, aunque esté formado por una sola.

La función σ_i^2 es de la forma: $\sigma_i^2(x,y) = \frac{(x+y) \cdot (x+y+1)}{2} + y$.

P.ej., el número asociado al siguiente programa P :

```

while  $x_2 \neq 0$  do
   $x_1 := x_1 + 1$ 
   $x_2 := x_2 - 1$ 
od
  
```

es:

$$\left. \begin{aligned}
 \text{cód}(x_1 := x_1 + 1) &= 5 * \sigma_1^2(0, 0) + 2 = 2 \\
 \text{cód}(x_2 := x_2 - 1) &= 5 * \sigma_1^2(1, 1) + 3 = 23
 \end{aligned} \right\} S$$

$$\text{cód}(S) = \sigma_1^2(1, \sigma_1^2(23, 2)) = \sigma_1^2(1, 327) = 54283$$

$$\text{cód}(\text{while } x_2 \neq 0 \text{ do } S \text{ od}) = 5 * \sigma_1^2(1, 54283) + 4 = 7367288769$$

$$\text{cód}(P) = \sigma_1^2(0, 7367288769) = 27138471913967700834$$

Se pide:

- a) Construir la gramática que reconozca los programas escritos en lenguaje SAL.
- b) Construir el programa LEX-YACC que dado un programa SAL, devuelva su número natural asociado.

NOTA: Suponer que el tipo **natural** con las siguientes operaciones definidas:

```
/* Funcion que asigna a un natural un valor int. */  
void asignar(natural * l_valor; unsigned int r_valor);  
/* Funcion que asigna a un natural la suma de otros dos. */  
void sumar(natural * l_valor, natural r1_valor, natural r2_valor)  
/* Funcion que asigna a un natural el producto de otros dos. */  
void multiplicar(natural * l_valor, natural r1_valor, natural r2_valor)  
/*Funcion que asigna a un natural su propio valor dividido entre 2.*/  
void dividir2(natural * l_valor)  
/* Operacion de visualizacion de un numero natural. */  
void visualizar(natural r_valor)
```

c) Para evitar la obtención de números innecesariamente grandes, podemos efectuar una transformación entre las variables, de manera que se renombran las variables con objeto de que no haya variables no utilizadas. P.ej:

```
x327 := 0  
while x24 ≠ 0 do  
    x327 := x327 + 1  
    x24 := x24 ÷ 1  
    x24 := x24 ÷ 1  
od  
x1 := x327
```

podría transformarse en algo así como:

```
x1 := 0  
while x2 ≠ 0 do  
    x1 := x1 + 1  
    x2 := x2 ÷ 1  
    x2 := x2 ÷ 1  
od  
x3 := x1
```

donde se ha sustituido el nombre de variable x_{327} por el nombre x_1 , el nombre x_{24} por el de x_2 , y el nombre x_1 por el de x_3 , lo cual permitirá que, teniendo un programa casi equivalente, el número natural asociado sea más pequeño.

Se pide explicar textualmente qué modificaciones serían necesarias en el analizador anterior para poder efectuar estas transformaciones, indicando en todo caso las características (estructura y operaciones) que debería poseer la tabla de símbolos para este propósito.