Tarea Programada Número 1 (parte de prolog)

En esta tarea usted debe hacer dos programas en prolog, el primero un problema de satisfacción de restricciones típico de los resueltos utilizando Prolog, el segundo un pequeño interprete de un lenguaje inventado que llamaremos simpl (simple).

Problema Número 1.
Ud debe hacer un programa en prolog que pueda resolver eficientemente un tablero de sudoku. En el sudoku se tiene una cuadrícula de 9x9, en la cual algunos de los recuadros de la cuadrícula contienen valores númericos ya pre-establecidos, los que se encuentran en blanco deben ser llenados por el algoritmo para resolver el problema. La restricción del juego es que los valores que se pueden poner van el 1 al 9, y no se deben repetir en ninguna fila, columna, o cuadrante de 3x3 (hay 9 cuadrantes).

Su programa debe recibir como entrada una lista de triadas, de la forma [[3,7,2], [5,1,9] …], tal que cada triada corresponde a un recuadro dentro de la cuadrícula que ya contiene un valor. Por ejemplo, para el caso de la lista anterior, [3,7,2] significa que el recuadro en la fila 3, columna 7, contiene el valor de 2, y el [5,1,9] indica que el recuadro en la fila 5, columna 1, contiene el valor de 9.

El programa debe desplegar el tablero de sudoku completamente lleno, cumpliendo con las restricciones del problema.

Problema Número 2.
En la página del curso del tecDigital, en la carpeta de simpl, se encuentran cuatro programas de Prolog, estos programas implementan un diminuto interprete de un lenguaje de programación. Este lenguaje tiene ciertas restricciones:

1. Sólo permite variables con valor numérico.
2. El if-then-else y la recursión son las únicas estructuras de control.
3. Es completamente funcional.
4. Solo permite comparar valores en la condición de los if-then-else, no tiene and, or, not …

A pesar de esto el lenguaje puede ejecutar programas pequeños como el fibonacci, factorial, y otros cálculos numéricos.

La gramática del lenguaje es la siguiente:

simpl	→	definición *simpleResto*
simplResto	→	**;** *simpl* | **nada**
definición	→	<**identificador**> = *expresión*
expresión	→	*let* | *lambda* | *if* | *fórmula*
fórmula	→	*término* | *fórmula* **+** *término* | *fórmula* **-** *término*
término	→	*factor* | *término* ***** *factor* | *término* **/** *factorial*
factor	→	<**número**> | **(** *expresión* **)** | *valor*
valor	→	*identificador paramList*
paramList	→	**nada** | **(** *params* **)**
params	→	*expresión restoParams*
restoParams	→	**nada** | **,** *params*
lambda	→	**fun (** *identList* **) :** *expresión*
identList	→	*identificador* [**,** *identList*]

if → **if** *condición* **then** *expresión* **else** *expresión*
let → **let** *simpl* **in** *expresión*
condición → *expresión op expresión*
op → < | = | >

Usted debe

1. Dado el siguiente programa de simpl.

```
x = 1;
f = fun(y) : x + 1;
g = fun(a) : let x = 2 in f(a)
```

¿A qué valor evalua la expresión g(3)? ¿A qué tipo de binding corresponde esto? ¿Dinámico o estático? ¿En qué parte del código es que se esta escogiendo el valor del x=1 ó x=2?

2. Agregar al simpl la posibilidad de tener condiciones booleanas en las expresiones de los if. Esto es
   2.1 Agregar las constantes **true** y **false** al lenguaje el tokenizador (tokenizer.pl).
   2.2 Reconocer las constantes **true** y **false** en el intérprete (interpreter.pl).
   2.3 Agregar los operadores **and**, **or** y **not** al lenguaje tokenizador (tokenizer.pl).
   2.4 Agregar el reconocimiento de las expresiones booleanas al parser (simple.pl).
   2.5 Reconocer la ejecución correcta de and or y not en el intérpreter (interpreter.pl).

Fecha de Entrega: 6 de Octubre.