Práctica 1 (Programación Lógica Pura)

Profesor responsable: Miguel García Remesal

1. Run-length encoding modificado (5 puntos)

Se pide al alumno implementar el algoritmo de compresión run-length encoding modificado. Este algoritmo toma como entrada una cadena de símbolos representada como una lista de constantes, y devuelve una lista de estructuras `rlec(Symbol,Count)` que denotan la aparición consecutiva de `Count` (número en notación de Peano) símbolos `Symbol` en la entrada. Obviamente, el orden de las estructuras `rlec/2` debe respetar el orden de aparición de los símbolos en la lista original. Si un elemento apareciese solamente una vez entre otros dos símbolos, este símbolo se copiará tal cual a la lista resultado, sin necesidad de estar encapsulado en una estructura de tipo `rlec(Symbol,Count)`. Se pide al alumno programar el predicado `comprimir/2` con cabecera `comprimir(LO,LC)`, donde `LO` y `LC` representan la lista de símbolos original y comprimida respectivamente. Debe programarse también el predicado `descomprimir/2` con cabecera `descomprimir(LC,LO)`, que realiza la operación inversa al predicado de compresión.

Ejemplo:

```
?- comprimir([a,a,a,a,a,a,a,b,b,b,b,b,c,c,d,d,d,d,e,f,f,f,
f,f,g,e,a,a,aa,b],C).

C = [rlec(a,s(s(s(s(s(s(s(0)))))))),rlec(b,s(s(s(s(s(0))))
)),rlec(c,s(s(0))),rlec(d,s(s(s(s(0))))),e,rlec(f,s(s(s(s(
s(0)))))),g,e,rlec(a,s(s(0))),aa,b] ? ;

no


?- descomprimir([rlec(a,s(s(s(s(s(s(s(0))))))),rlec(b,s(s(
s(s(s(s(0)))))),rlec(c,s(s(0))),rlec(d,s(s(s(s(0))))),e,rlec
(f,s(s(s(s(s(0)))))),g,e,rlec(a,s(s(0))),aa,b], D).

D = [a,a,a,a,a,a,a,b,b,b,b,b,c,c,d,d,d,d,e,f,f,f,f,f,g,e,a
,a,aa,b] ? ;

no
```

2. Exploración de árboles n-arios (5 puntos)

Se tiene la siguiente representación de árboles n-arios de números naturales. Las hojas del árbol se representan mediante la estructura `hoja/1`, donde

el argumento es el número natural almacenado en dicho nodo hoja. El resto de nodos (no hoja) se representan utilizando la estructura `nodo/2` con cabecera `nodo(N,Hijos)` donde `N` es el número natural almacenado en dicho nodo e `Hijos` es una lista no vacía de árboles n-arios de números naturales. Se pide programar el predicado `menores/2` con cabecera `menores(Arbol, Max)` que será cierto si dado el árbol n-ario `Arbol` (que puede contener variables en los nodos), los números naturales almacenados en el árbol son todos menores o iguales a `Max`, que es un número natural en notación de Peano. En caso de que el árbol contenga variables en los nodos, estas deberán instanciarse con valores en dicho rango. Se pide también implementar el predicado `suma/2` con cabecera `suma(Arbol, Suma)` que dado el árbol n-ario `Arbol` calcula la suma de todos los números naturales contenidos en el árbol y la unifica con la variable `Suma`.

Ejemplos:

```
?- menores(nodo(s(s(0)),[nodo(s(0),[hoja(0),hoja(s(s(0))),nodo(
s(s(s(0))),[hoja(s(0))])]),hoja(s(s(0))),hoja(0),hoja(s(s(0)))]
),s(s(s(0)))).

yes


?- menores(nodo(s(s(0)),[nodo(s(0),[hoja(0),hoja(X),nodo(s(s(s(0
))),[hoja(s(0))])]),hoja(Y),hoja(0),hoja(s(s(0)))]),s(s(s(0)))).

X = 0, Y = 0 ? ;
X = 0, Y = s(0) ? ;
X = 0, Y = s(s(0)) ? ;
X = 0, Y = s(s(s(0))) ? ;
X = s(0), Y = 0 ? ;
X = s(0), Y = s(0) ? ;
X = s(0), Y = s(s(0)) ? ;
X = s(0), Y = s(s(s(0))) ? ;
X = s(s(0)), Y = 0 ? ;
X = s(s(0)), Y = s(0) ? ;
X = s(s(0)), Y = s(s(0)) ? ;
X = s(s(0)), Y = s(s(s(0))) ? ;
X = s(s(s(0))), Y = 0 ? ;
X = s(s(s(0))), Y = s(0) ? ;
X = s(s(s(0))), Y = s(s(0)) ? ;
X = s(s(s(0))), Y = s(s(s(0))) ? ;

no


?- suma(nodo(s(s(0)),[nodo(s(0),
[hoja(0),hoja(s(s(0))),nodo(s(s(s(0))),[hoja(s(0))])]),hoja(s(s(
0))),hoja(0),hoja(s(s(0)))]),Suma).
```

```
Suma = s(s(s(s(s(s(s(s(s(s(0)))))))))))) ? ;

no
```

Normas

La fecha límite de entrega de la práctica es el **Jueves 23 de Noviembre de 2016 a las 15:30 horas**. Esta práctica puede hacerse en grupos de 3 personas, siendo posible también hacerla de forma individual. Uno de los alumnos del grupo asumirá el rol de portavoz del grupo, y será el encargado de subir la práctica a Moodle. Es importante que UNICAMENTE EL PORTAVOZ del grupo suba la práctica a Moodle. Una vez asumida la portavocía de grupo, esta se mantiene para las prácticas subsiguientes.

El código debe entregarse en un fichero denominado exactamente `pract1.pl` donde deben estar implementados todos los predicados requeridos en el enunciado. Dado que esta práctica será corregida utilizando un corrector automático, es OBLIGATORIO que el functor, aridad y orden de los argumentos DE TODOS LOS PREDICADOS mencionados en el enunciado sea IDÉNTICO al indicado en el enunciado. Si esto no se hace así, el corrector automático rechazará la práctica y esta se calificará como suspenso. El fichero pract1.pl debe estar DEBIDAMENTE COMENTADO, explicando la semántica de cada uno de los predicados desarrollados en la práctica.

Dentro del fichero pract1.pl se indicará quienes son los componentes del grupo de prácticas mediante la inclusión de tantos hechos de tipo `alumno_prode/4` como alumnos haya en el grupo. Estas estructuras, con cabecera:

```
alumno_prode(Apellido1,Apellido2,Nombre,NumMatricula)
```

indican al corrector automático los datos de cada componente del grupo. Por ejemplo, si uno de los componentes del grupo se llama Ignacio Javier García Lombardía, con número de matrícula D16025, entonces será necesario incluir el hecho:

```
alumno_prode('Garcia', 'Lombardia', 'Ignacio Javier', 'D160125')
```

en la base de hechos del programa. Es SUMAMENTE IMPORTANTE seguir este paso, ya que de no hacerse correctamente, el corrector automático no podrá discernir quien es el autor de la práctica, y por tanto, la rechazará y como consecuencia la calificará como suspensa. Es responsabilidad de cada uno de los componentes del grupo el asegurarse de que estos hechos están correctamente declarados antes de entregar la práctica.

Se recuerda al alumno que no es posible utilizar ningún recurso no puro. El uso de este tipo de recursos acarrea un decremento sustancial de la nota e incluso el suspenso de la práctica dependiendo de los recursos utilizados.