Programación declarativa **(29–Noviembre–2004)**

Apellidos:

Nombre:

Nota: *En la evaluación de los ejercicios se tendrá en cuenta la simplicidad y la eficiencia de las soluciones.*

Ejercicio 1 [2 puntos] Definir la relación `lista(+N,-L)` que se verifique si `L` es la lista de longitud `N` cuyos elementos son `N`. Por ejemplo,

```
?- lista(3,L).
L = [3, 3, 3]
```

.....

Solución: La definición de `lista` es

```
lista(N,L) :-
   lista_aux(N,N,L).
```

donde `lista_aux(+N,+M,-L)` se verifica si `L` es la lista de longitud `M` cuyos elementos son `N`. Por ejemplo,

```
?- lista_aux(5,4,L).
L = [5, 5, 5, 5]
```

y se define por

```
lista_aux(_,0,[]).
lista_aux(N,M,[N|L]) :-
   M > 0,
   M1 is M-1,
   lista_aux(N,M1,L).
```

Ejercicio 2 [3 puntos] Definir las siguientes relaciones:

- `menor_divisor_propio(+N,?X)` que se verifique si `X` es el menor divisor propio de `N`. Por ejemplo,

  ```
  ?- menor_divisor_propio(30,X).
  X = 2
  ?- menor_divisor_propio(3,X).
  X = 3
  ```

- `factorización(+N,-L)` que se verifique si `L` es la lista correspondiente a la descomposición del número `N` en factores primos (se considera los elementos de `L` están ordenados de manera creciente). Por ejemplo,

  ```
  ?- factorización(12,L).
  L = [2, 2, 3] ;
  No
  ```

.....

Solución: La definición de `menor_divisor_propio` es

```
menor_divisor_propio(N,X) :-
   between(2,N,X),
   N mod X =:= 0, !.
```

La definición de `factorización` es

```
factorización(1,[]).
factorización(N,[X|L]) :-
   N > 1,
   menor_divisor_propio(N,X),
   N1 is N/X,
   factorización(N1,L).
```

Ejercicio 3 [2 puntos] Definir la relación `crecimientos(+L1,-L2)` que se verifique si `L2` es la lista correspondientes a los crecimientos de la lista numérica `L1`; es decir, entre cada par de elementos consecutivos `X` e `Y` de `L1` coloca el signo `+` si `X < Y` e y signo `-` en caso contrario. Por ejemplo,

```
?- crecimientos([1,3,2,2,5,3],L).
L = [1, +, 3, -, 2, -, 2, +, 5, -]
```

Dar una definición sin corte y otra con corte.

.....

Solución: La definición de `crecimientos` sin usar corte es

```
crecimientos([_],[]).
crecimientos([X,Y|L1],[X,+|L2]) :-
   X < Y,
   crecimientos([Y|L1],L2).
crecimientos([X,Y|L1],[X,-|L2]) :-
   X >= Y,
   crecimientos([Y|L1],L2).
```

La definición de `crecimientos` usando corte es

```
crecimientos([_],[]).
crecimientos([X,Y|L1],[X,+|L2]) :-
   X < Y, !,
   crecimientos([Y|L1],L2).
crecimientos([X,Y|L1],[X,-|L2]) :-
   % X >= Y,
   crecimientos([Y|L1],L2).
```

Ejercicio 4 [3 puntos] Definir las siguientes relaciones:

- `longitud(+P,-N)` que se verifique si `N` es la longitud de la palabra `P`. Por ejemplo,

  ```
  ?- longitud(ana,N).
  N = 3
  ```

- `palabra_maximal(+L,-P)` que se verifique si `P` es una palabra maximal (es decir, de máxima longitud) de la lista de palabras `L`. Por ejemplo,

  ```
  ?- palabra_maximal([eva,y,ana,se,van],P).
  P = eva ;
  P = ana ;
  P = van ;
  No
  ```

- `palabras_maximales(+L1,-L2)` que se verifique si `L2` es la lista de las palabras maximales de la lista de palabras `L1`. Por ejemplo,

```
?- palabras_maximales([eva,y,ana,se,van],L).
L = [eva, ana, van]
```

.....

Solución: La definición de `longitud` es

```
longitud(P,N) :-
   name(P,L),
   length(L,N).
```

La definición de `palabra_maximal` es

```
palabra_maximal(L,P) :-
   select(P,L,R),
   longitud(P,N),
   not((member(P1,R), longitud(P1,N1), N < N1)).
```

La definición de `palabras_maximales` es

```
palabras_maximales(L1,L2) :-
   findall(P,palabra_maximal(L1,P),L2).
```
