# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
# FACULTAD DE CIENCIAS
# LÓGICA COMPUTACIONAL

Profesora: ESTEFANÍA PRIETOS LARIOS
Ayudante: HERNÁNDEZ OLVERA MAURICIO E.
Ayudante Lab: SALAZAR GONZÁLEZ EDWIN MAX

Fecha de Entrega: Jueves 11 de Abril, 2019. 23:59

PRÁCTICA 2. Parte 3 ☺

PROLOG

## OBJETIVOS

- Reforzar los conceptos básicos del lenguaje lógico PROLOG.
- Se espera que el alumno ponga en práctica los conceptos sobre manejo de listas, árboles y gráficas en *prolog*.

## INSTRUCCIONES

Su misión, si deciden aceptarla es:
Crear el archivo *Practica2p3.pl* y resolver al menos 10 de los siguientes ejercicios.

## EJERCICIOS

1.- Crea la regla **range**
Recibe tres parámetros: M, N, L
La regla será satisfactoria cuando se cumpla la propiedad de que L contiene el rango entre M y N.

Nota: Se sugiere el uso de IF en prolog y la regla concatena cuando N<M.

```
| ?- range (2, 8, Z).
Z = [2,3,4,5,6,7,8] ? ;
no

| ?- range (8, 2, Z).
Z = [8,7,6,5,4,3,2] ? ;
no
```

2.- Crea la regla **rotate**
Recibe tres parámetros: L, N, A.
La regla será satisfactoria cuando se roten los primeros N elementos de la lista L.
La lista resultante se guardará en la variable A.

```
| ?- rotate ([1,2,3,4,5], 1, A).
A = [2,3,4,5,1] ? ;
no
```

3.-Crea la regla **compress**
Recibe dos parámetros: L, Lc
La regla será satisfactoria cuando se cumpla la propiedad de que Lc tiene la compresión de los elementos de la lista L.

```
| ?- compress([G,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,l], J).
J = [G,o,l] ? ;
no
```

4.- Crea la regla **c_by_group**
Recibe dos parámetros: L, LA
La regla será satisfactoria cuando se cumpla la propiedad de que LA tiene el agrupamiento de los elementos de la lista L

```
| ?- c_by_group([G,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,l,_,G,o,o,o,o,l], J).
J = [[G],[o,o,o,o,o,o,o,o,o,o,o,o,o,o,o,o],[l],[_],[G],[o,o,o,o],[l]] ? ;
no
```

5.- Crea la regla **e12**
Recibe dos parámetros: L, LAC
La regla será satisfactoria cuando se cumpla la propiedad de que LAC tiene el agrupamiento compreso de los elementos de la lista L

```
| ?- e12([G,o,o,o,o,o,o,o,o,o,o,o,o,l], K).
K = [[1,G],[12,o],[1,l] ? ;
no
```

6 .- Crea la regla **degree**
Recibe tres parámetros: G,N,D
La regla será satisfactoria cuando se cumpla la propiedad de que D es el grado del nodo N en la gráfica G.

```
| ?- degree([(a,b),(c,a),(a,d),(d,c),(e,c),], a, R).
R = 3 ? ;
no
```

7 .- Crea la regla **neighbors**
Recibe tres parámetros: G,N,NL
La regla será satisfactoria cuando se cumpla la propiedad de que NL es la lista de vecinos del nodo N en la gráfica G.

```
| ?- neighbors([(a,b),(c,a),(a,d),(d,c),(e,c),], d, N).
N = [a,c] ? ;
no
```

8 .- Crea la regla **path**
Recibe cuatro parámetros: G, O, D, P
La regla será satisfactoria cuando se cumpla la propiedad de que P es el camino que existe del nodo origen (O) al nodo destino (D) en la gráfica G.

```
| ?- path([(a,b),(c,a),(a,d),(d,c),(e,c),], a, e, P).
P = [a,d,c,e] ? ;
P = [a,c,e] ? ;
no
```

9 .- Crea la regla **preorder**
Recibe dos parámetros: T, P
La regla será satisfactoria cuando se cumpla la propiedad de que P es el recorrido en pre-orden en el árbol T.

```
| ?- preorder(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3,nil,nil)), V).
V = [1,2,4,5,3]
yes
```

10 .- Crea la regla **num_leaves**
Recibe dos parámetros: T, N
La regla será satisfactoria cuando se cumpla la propiedad de que N es el número de hojas del árbol T.

Se recomienda el uso de **!** para evitar tener resultados repetidos.

```
| ?- num_leaves(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3,nil,nil)), U).
U = 3
yes
```

11 .- Crea la regla **list_leaves**
Recibe dos parámetros: T, L
La regla será satisfactoria cuando se cumpla la propiedad de que L es la lista de hojas del árbol T.

Se recomienda el uso de **!** para evitar tener resultados repetidos.

```
| ?- list_leaves(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3,nil,nil)), W).
W = [4,5,3]
yes
```

12 .- Crea la regla **is_symetric**
Recibe un parámetro: T
La regla será satisfactoria cuando se cumpla la propiedad de que el árbol T es simétrico.

Ojo: Recordar que es simétrico en cuanto a la estructura, no a sus elementos.

```
| ?- is_symetric(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3,nil,nil))).
no

| ?- is_symetric(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3,t(6,nil,nil),nil))).
no

| ?- is_symetric(t(1, t(2, t(4,nil,nil), t(5,nil,nil)), t(3, t(6,nil,nil), t(7, nil, nil)))).
si
```

## REGLAS DEL JUEGO:

- ✓ Respetar las firmas de las reglas.
- ✓ **Todas** las firmas deberán estar documentadas.
- ✓ La legibilidad y documentación tendrá un impacto sobre la calificación de la práctica.
- ✓ La práctica se podrá realizar en equipos de a lo más 3 personas.
- × Cualquier plagio de prácticas será evaluado con 0, sin hacer indagaciones.
- × Cualquier copia de internet, sin entender el código será 0 sobre **la práctica.** Así se haya copiado solo en un ejercicio.
- × Cualquier práctica entregada posterior a la fecha límite no será tomada en cuenta.

Se deberá contar con un directorio cuyo nombre sea Practica2. Dentro del directorio se debe tener:

- README.txt, donde se incluya número de cuenta y comentarios sobre la práctica.
- Practica2p3.pl, script requerido para ésta práctica.

Comprimir el directorio con el formato ApellidoNombreP2p3. Comprimir con extensión .tar.gz o .zip

Si la práctica se realiza en equipo, agregar el nombre y número de cuenta de los integrantes en el README.txt

Enviar la práctica al correo ciclomax9@ciencias.unam.mx con el asunto [LC-Apellido-Nombre-P2p3].

*Suerte ☺*

*"Si la depuración es el proceso de eliminar errores, entonces la programación debe ser el proceso de introducirlos"*
*-- Edsger W. Dijkstra*