Lógico: Ejercicios Otros

En estos ejercicios se empiezan a mezclar cosas que vimos en la materia con otras que no vimos, o no pusimos mucho énfasis; están pensados sobre todo para los que quieran seguir aprendiendo sobre programación lógica.

Ejercicio 1 – Mensajes de Texto

Se pide realizar un programa en prolog con la siguiente funcionalidad

a) Se cuenta con la siguiente base de conocimiento

```
tel(1,"!?@").
tel(2,"abc").
tel(3,"def").
tel(4,"ghi").
tel(5,"jkl").
tel(6,"mno").
tel(7,"pqrs").
tel(8,"tuv").
tel(9,"wxyz").
tel(0," ").
```

Implementar el predicado nrosPara/2, según esta especificación y ejemplos de consultas:

% nrosPara(String,ListaDeNumeros)

% String representa una palabra o frase y ListaDeNumeros las teclas asociadas al teléfono para poder representarla

```
?- nrosPara("hola",Numeros).

Numeros = [4, 6, 5, 2] ;
No

?- nrosPara(“hola como estas?”, Numeros).

Numeros = [4, 6, 5, 2, 0, 2, 6, 6, 6|...] [write]

% presionando la tecla W muestra la lista completa

Numeros = [4, 6, 5, 2, 0, 2, 6, 6, 6, 0, 3, 7, 8, 2, 7, 2] ;
No
```

El predicado nrosPara/2 debe ser inversible

```
?- nrosPara(Letras,[4, 6, 5, 2]).
Letras = [103, 109, 106, 97] ;
Letras = [103, 109, 106, 98] ;
Letras = [103, 109, 106, 99] ;
Letras = [103, 109, 107, 97] ;
Letras = [103, 109, 107, 98] ;
Letras = [103, 109, 107, 99] ;
Letras = [103, 109, 108, 97] ;
etc.
```

Tener en cuenta que SWI Prolog considera a un String como la lista de sus números ASCII

asociados.
En las consultas, pueden usar el predicado string_to_list para transformar la lista de ASCII a un string legible

?- nrosPara(Letras,[4, 6, 5, 2]), string_to_list(ResultadoMostrable,,Letras).

Lo malo de string_to_list es que no es inversible =(, por eso no conviene usarlo dentro de nrosPara que sí queremos que sea inversible.

b) Hacer el predicado nroGratis que relaciona una empresa con su número gratuito (el número gratuito tiene el formato 0800)

?- nroGratis(“skip”, Numeros).
Numeros = [0, 8, 0, 0, 7, 5, 4, 7] ;
No

c) Ahora se cambia nuestra base de conocimiento

```
tel(1,'!'). tel(1,'?'). tel(1,'@').
tel(2,'a').tel(2,'b').tel(2,'c').
tel(3,'d').tel(3,'e').tel(3,'f').
tel(4,'g').tel(4,'h').tel(4,'i').
tel(5,'j').tel(5,'k').tel(5,'l').
tel(6,'m').tel(6,'n').tel(6,'o').
tel(7,'p').tel(7,'q').tel(7,'r').tel(7,'s').
tel(8,'t').tel(8,'u').tel(8,'v').
tel(9,'w').tel(9,'x').tel(9,'y').tel(9,'z').
tel(0,' ').
```

Desarrollar nuevamente los predicados nrosPara/2 y nroGratis/2
Ayuda: utilizar los predicados para manejo de átomos atom_chars/2 y atom_concat/3

Ejercicio 2 - Manejo de Matrices en Lógico

Una matriz puede verse como una lista de listas, por ejemplo:

```
[[1,4], [2, 6]]     [[3, 6, 9], [5, 4, 1], [8, 2, 10]]
```

puede verse como las matrices cuadradas:

1	4
2	6

3	6	9
5	4	1
8	2	10

Ahora, veamos un poco más atentamente la segunda matriz. Podemos ver que es una matriz con varias propiedades particulares:

- Propiedad uno: Si intercalamos operaciones de sumas y restas entre los números de sus *filas* vemos que se forman operaciones aritméticas válidas:

3	+	6	=	9
5	-	4	=	1
8	+	2	=	10

- Propiedad dos: Si intercalamos operaciones de sumas y restas entre los números de sus *columnas* vemos que se forman operaciones aritméticas válidas:

3	6	9
+	-	+
5	4	1
=	=	=
8	2	10

Otra forma de expresar esta propiedad es decir que la **transpuesta** de la matriz cumple con la propiedad uno. (Las operaciones aritméticas no tiene por que ser las mismas)

3	+	5	=	8
6	-	4	=	2
9	+	1	=	10

- Propiedad tres: No se repiten los números usados en la matriz

A este tipo de matrices las llamaremos **Cuadrados Aritméticos**.

- Trabajaremos solo con matrices de 3X3, y con números en el rango de 1 a 10 inclusive
- Seria interesante poder hacer un programa en prolog que me permita comprobar/generar cuadrados aritméticos. Esto lo vamos a tratar de hacer a lo largo de la ejecitación.

Planteo General, con enfoque top-down

Pensemos primero en las propiedades que sabemos y como las podemos relacionar en un solo predicado:
Podemos suponer que exite el predicado **cumpleOperaciones(M)** verifica la propiedad uno.
Tambien que tenemos un predicado **transpuesta(M,MT)**, que podemos expresar la propiedad dos.
Y con el predicado **cumpleUnicidad(M)** expresamos la propiedad tres.

Con esto, podemos armar el predicado cuadradoAritmetico:

```
cuadrado_aritmetico(M):-
cumpleOperaciones(M),cumpleUnicidad(M),transpuesta(M,MT),cumpleOperaciones(MT).
```

El predicado cumpleUnicidad(M)

Como podemos determinar que los elementos de una matriz no se repiten?
Tip 1: es fácil determinar si los elementos de una lista no se repiten, si tuvieramos la lista aplanada de la matriz....
Tip 2: dijimos que una matriz es una lista de listas, recordá que en una definición recursiva sobre listas, parto de la definición para la cola, en este caso puedo suponer que sé cuál es el resultado de aplanar la cola...

El predicado transpuesta(M,MT)

Como podemos relacionar una matriz con su transpuesta?
Si pensamos que una matriz es una lista formada por las filas,
suena natural pensar que la trasnpuesta es la *lista fomrmada por las columnas* de la matriz original.
Entonces, tenemos que pensar en una forma de obtener las columnas de una matriz de una forma similar a como prolog nos permite obtener los elementos de una lista. O sea, separo la obtención de la primer columna del resto:

primerColumna([[3, 6, 9], [5, 4, 1], [8, 2, 10]],[3, 5, 8],[[6, 9], [4, 1], [2, 10]]).

primerColumna relaciona una matriz en su primer columna y el resto.

Tip: Parece complicado, pero se aclara si pensamos que la primer columna es la lista de los primeros elementos de cada fila; en particular, el primer elemento de la primer fila, es tambien el primer elemento de la primer columna .
Una ver realizado el predicado primerColumna, es más facil hacer el transpuesta.

Tip 1: La transpuesta de una matriz tendrá
- como primer fila: la primer columna de la matriz.
- como resto de las filas: el resultado de transponer el resto.

Tip 2: Con lo anterior más pruebas en papel debería salir fácil.

El predicado cumpleOperaciones(M)

Este el el gran predicado del problema, pero ya hemos reducido mucho su alcance. Solo debe determinar si:

3	6	9
5	4	1
8	2	10

Cumple con

3	**?**	6	=	9
5	**?**	4	=	1
8	**?**	2	=	10

Donde **?** puede ser en cada caso una suma o una resta.
Como estamos trabajando con matrices de 3X3 podemos asumir que la entrada es una lista de exactamenten 3 elementos.
Cada uno de estos debe de cumplir una operación. Entonces, qué mejor que armar un predicado operacion que relacione tres números si están ligados por alguna operación. Si recuerdo que los números válidos son solo los que estan entre 1 y 10 ambos inclusive, este predicado se puede hacer inversible.

Generación de cuadrados aritméticos

Hacer el predicado cuadrado_aritmetico(X), y verificar que es inversible (o sea que si no ligo el argumento en la consulta, genera los cuadrados).

Cuántos cuadrados aritméticos existen?

Hacer un predicado que permita determinar la cantidad de cuadrados aritméticos posibles a partir de los predicados construidos hasta ahora.