PRÁCTICA PROGRAMACIÓN LÓGICA (PROLOG)

Ejercicio 1

Dado un programa con hechos del tipo `persona(Nombre, Sexo, Edad)` y `padreDe(Nombre, Nombre)`, construir las siguientes reglas:

- `madreMasDe45(Nombre)` : determina si la persona es madre y tiene 45 años o más.
- `madreAMasDe35(Nombre)` : determina si la persona fue madre cuando tenía 35 años o más.
- `hijoDePadreMasDe50(Nombre)` : determina si una persona es hija de un padre que tiene 50 años o más.

Ejercicio 2

Dado un programa con hechos del tipo `jefe(symbol, symbol)`, que indican una relación de jerarquía, construir la regla `acata(symbol, symbol)` que indique si una persona puede dar órdenes a otra, según la cadena de mando.

Ejercicio 3

Dado un programa con hechos del tipo `primario(Color)` y `formadoPor(Color, Color, Color)`, construir las siguientes reglas:

- `secundario(X)` : determina si X es secundario, es decir está formado por dos colores primarios.
- `complementarios(X,Y)` : un color `X` es complementario de un color primario `Y`, cuando `X` se forma con dos colores primarios entre los cuales `Y` no se encuentra. Debe utilizar la regla anterior.

Ejercicio 4

En revistas y diarios suelen aparecer problemas como el siguiente:

> Instrucciones: colocar en los cuadros en blanco las cifras (de 0 a 9) correspondientes para que, vertical y horizontalmente, efectuando las operaciones indicadas, den la solución reseñada.

Para resolver este problema se pide codificar el predicado `result(A1,A2,A3,B1,B3,C1,C2,C3)`, que dé el o los resultados factibles para:

```
(A1 * A2) * A3 = 6
(B1 * 5) - B3 = 1
(C1 \ C2) * C3 = 4
(A1 - B1) * C1 = 7
(A2 + 5) / C2 = 1
(A3 + B3) - C3 = 7
```

Ejercicio 4 - Tratamiento de Listas.

Codificar las reglas:

- Pertenece/member/esta
- Longitud
- Agregar un elemento a la cabeza
- Agregar un elemento en una posición determinada
- Concatenar dos listas

Ejercicio 5 - Escribir en celulares

Conociendo la relación existente en los teclados de los teléfonos celulares de los números con las letras,

Número	Letras
2	a b c
3	d e f
4	g h i
5	j k l
6	m n o
7	p q r s
8	t u v
9	w x y z

y sabiendo que en la base de conocimiento, aparece sólo una vez cada número del teclado, se pide:

1. Codificar el predicado `conversion` (y sus auxiliares) tal que dada una lista de números genere las listas de letras factibles.
2. Con tu resolución, qué sucede si la consulta es `conversion(X, ["a", "e"])` ?

Ejemplos:

```
conversion ([2, 3], X)
X = [‘a’, ‘d’]
X = [‘b’, ‘d’]
X = [‘c’, ‘d’]
X = [‘a’, ‘e’]
X = [‘b’, ‘e’]
X = [‘c’, ‘e’]
X = [‘a’, ‘f’]
X = [‘b’, ‘f’]
X = [‘c’, ‘f’]
false
```

```
conversion ([2, 3], [‘a’, ‘e’])
True
```

Ejercicio 6

El servicio de Web Mail de un portal calcula la contraseña de los usuarios en base al número de usuario (el cual se tiene como una lista de dígitos). La contraseña es de tres números y los calcula de la siguiente forma:

- Primer dígito: factorial del primer dígito del número de usuario.
- Segundo dígito: longitud del número de usuario.
- Tercer dígito: suma de los dígitos de las posiciones pares del número de usuario.

Escribir un programa Prolog que en base a un número de usuario calcule la contraseña.

Ejemplo:

```
generar_contraseña([2, 3, 7, 1, 6, 8], Contraseña)
Contrasenia = [2, 6, 12]

generar_contraseña([2, 3, 7, 1, 6, 8], [3, 5, 7])
False
```

Ejercicio 7 - Mejor vuelto

Definir un programa Prolog que permita determinar el mejor vuelto a dar a una persona que abona una suma de dinero. Se debe tener en cuenta que:

- Los montos a abonar son enteros.
- El mejor vuelto es aquél que contiene la menor cantidad de billetes.
- Para dar vuelto se poseen billetes con las siguientes numeraciones `1, 2, 5, 10, 20, 50, 100, 200, 500 y 1000`.

El programa debe responder a consultas del tipo:

```
mejorVuelto(100, 86, Lista)
Lista = [10, 2, 2]
```

Ejercicio 8

Los auxiliares de contaduría de una empresa nos presentaron el siguiente problema:

> Diariamente depositan una cantidad variable de cheques. Oportunamente, el banco informa el monto total acreditado en la cuenta, sin detallar los importes que lo conforman, dándose la posibilidad de que sólo algunos cheques se hubieran acreditado.

1. Dada la necesidad de reconocer qué cheques se acreditaron, solicitan un programa que les permita conocer todas las posibilidades a partir de la consulta `acredPosibles(ImportesDeCheques, MontoAcred, ImportesPosibles)`.

Ejemplos:

```
acredPosibles([10, 12, 20, 30, 50, 60, 100], 282, X)
X = [10, 12, 20, 30, 50, 60, 100]


acredPosibles([10, 12, 20, 30, 50, 60, 100], 60, X)
X = [10, 20, 30]
X = [10, 50]
X = [60]
```

2. Definir una solución que permita además identificar a cada cheque.

Ejercicio 9

Una empresa que se dedica a la fabricación de muebles de madera solicita un sistema Prolog que le permita calcular sus costos de producción.

Para ello diseñó una base de conocimientos con hechos del tipo:

```
costos(ListaDeCostos)
requerimientos(Producto,ListaDeRequerimientos)
```

La lista de costos permite relacionar cada material con su costo por unidad; y la lista de requerimientos relaciona cada uno de los materiales requeridos con la cantidad de unidades necesarias de dicho material para construir el producto.

Construir el programa Prolog que responda a consultas del tipo `calcularCosto(Producto,Costo)` que determinan el costo total de producción de una unidad del producto indicado.

Se cuenta con:

```
costos([c(tabla1, 80), c(tabla2, 70), c(liston1, 20), c(liston2, 5),  c(tornillo, 1), c(clavo, 1)]).

requerimientos(mesa, [r(liston1, 4), r(tabla1, 10), r(liston2, 4),  r(tornillo, 50)]).
...
```

Ejercicio 10

Una facultad desea un sistema para administrar las electivas que les ofrece a sus alumnos. La información recolectada se volcó en hechos del tipo:

```
electivas(ListaDeMaterias).
datosMateria(NombreMateria, Area, PuntosQueOtorga).
```

donde la `ListaDeMaterias` contiene los nombres de las electivas que se dictan en la facultad, y el predicado `datosMateria` permite conocer de cada electiva el área temática y la cantidad de puntos que otorga cursarla. > Completar el programa Prolog que permite responder a consultas del tipo `combinarElectivas(PuntosNecesarios, AreasSeleccionadas, Electivas)` que permitan determinar las `Electivas` que puede cursar un alumno, para obtener los `PuntosNecesarios`, teniendo en cuenta las `AreasSeleccionadas`.

Se cuenta con:

```
electiva([e1,e2,e3,e4,e5,e6,e7,e8]).
datosMateria(e1,a1,5).
datosMateria(e2,a1,3).
...
```

Ejercicio 11

Una agencia de publicidad solicita un sistema Prolog para administrar los premios que otorga a los empleados que cumplen los objetivos fijados.

Para ello diseñó una base de conocimiento con hechos del tipo `empleadosDeLaAgencia(ListaDeEmpleados)` que permite establecer la ListaDeEmpleados que actualmente trabajan en la empresa; y `contratosDelMes(ListaDeContratos)` que permite determinar la `ListaDeContratos` logrados durante el mes, donde cada elemento de la lista tiene la forma `c(empleado, empresa, monto)`, indicando qué empleado consiguió el contrato, con qué empresa se firmó, y cuál es el monto del mismo.

Completar el programa Prolog que permita realizar consultas del tipo `premiar(CantMinimaContratos, MontoMinimoContratos, EmpleadosPremiados)` que permita determinar la lista de `EmpleadosPremiados`, teniendo en cuenta que los mismos deben haber logrado durante el mes una cantidad de contratos mayor o igual a la `CantMinimaContratos`, o bien el monto total de los contratos conseguidos debe superar el `MontoMinimoContratos`.

Se cuenta con:

```
empleadosDeLaAgencia([e1, e2, e3, e4, e5]).

contratosDelMes([c(e2,a,100), c(e3,f,300), c(e4,j,150), c(e2,h,200),
c(e1,d,180), c(e5,n,200),
c(e2,s,90), c(e4,i,400)]).
```

Ejemplos:

```
premiar(3, 500, X)
X = [e2,e4]

premiar(3, 200, X)
X = [e2, e3, e4]

premiar(4, 600, X)
X = []

premiar(6, 400, X)
X = [e4]
```

Ejercicio 12

Un nutricionista nos pide ayuda para armar un sistema que le permita generar las dietas que recomienda a sus pacientes, y nos brinda la información sobre comidas volcada en una base de conocimiento mediante hechos del tipo `opcionesPara(Comida, Opciones)` que permiten establecer la lista de `Opciones` para una determinada `Comida` (desayuno, almuerzo, etc), donde cada opcion está representada por un functor de la forma `c(nombreComida, calorias)`.

Completar el programa Prolog que permita realizar consultas del tipo `armarDieta(ListaComidas, MaximoDeCalorias, Opciones, Calorias)` que permitan determinar todas las combinaciones posibles de `Opciones`, teniendo en cuenta una opción de cada comida de las indicadas en `ListaComidas`, controlando que no se supere el `MaximoDeCalorias`, y cuántas `Calorias` implica esa combinación de opciones de comidas.

Se cuenta con:

```
opcionesPara(desayuno, [c(tostadas, 10), c(yogurth, 10), c(barraCereal, 15), c(fruta, 5)]).
opcionesPara(almuerzo, [c(bife, 80), c(filet, 50), c(fideos, 55), c(ensalada, 40)]).
opcionesPara(cena, [c(ravioles, 65), c(pizza, 55), c(pollo, 70), c(asado, 90)]).
```

Ejemplos:

```
armarDieta([desayuno, almuerzo, cena], 105, Opciones, Calorias)
Opciones=[c("tostadas",10),c("ensalada",40), c("pizza",55)], Calorias=105
Opciones=[c("yogurth",10), c("ensalada",40), c("pizza",55)], Calorias=105
Opciones=[c("fruta",5), c("ensalada",40), c("pizza",55)], Calorias=100


armarDieta([almuerzo, cena], 111, Opciones, Calorias)
Opciones=[c("filet",50), c("pizza",55)], Calorias=105
Opciones=[c("fideos",55), c("pizza",55)], Calorias=110
Opciones=[c("ensalada",40), c("ravioles",65)], Calorias=105
Opciones=[c("ensalada",40), c("pizza",55)], Calorias=95
Opciones=[c("ensalada",40), c("pollo",70)], Calorias=110
```

Ejercicio 13

Una compañía otorga puntos a sus clientes por los servicios utilizados, y canjea los mismos por regalos. La cantidad de `Puntos` asociados a cada `Servicio` se expresa mediante hechos del tipo `puntosPorServicio(Servicio, Puntos)` y el catálogo de regalos mediante un hecho del tipo `catalogo(Regalos)` donde `Regalos` es una lista de functores `r(regalo, puntos)`, que establece la cantidad de `puntos` necesarios para obtener un determinado `regalo`.

Se desea obtener un programa Prolog que permita realizar consultas del tipo `regalosPosibles(Servicios, Regalos, PuntosCanjeados)` tal que dada una lista de `Servicios` utilizados por un cliente, permita sugerir todas las combinaciones posibles de `Regalos` que el mismo podría canjear por los puntos correspondientes, indicando la cantidad de `PuntosCanjeados`, teniendo en cuenta que la compañía exige canjear como mínimo el 50% de los puntos acumulados.

Se cuenta con:

```
puntosPorServicio(seguro, 100).
puntosPorServicio(poliza, 130).
puntosPorServicio(cajaSeguridad, 30).
puntosPorServicio(tarjeta, 50).
puntosPorServicio(cuenta, 80).
puntosPorServicio(agenteBolsa, 100).
catalogo([r(paraguas, 30), r(reloj, 50), r(viaje, 300), r(spa, 200), r(cena, 100)]).
```

Ejemplos:

```
regalosPosibles([tarjeta, poliza], X, Y)
X=[r("paraguas",30),r("reloj",50),r("cena",100)], Y=180
X=[r("paraguas",30),r("cena",100)], Y=130
X=[r("reloj",50),r("cena",100)], Y=150
X=[r("cena",100)], Y=100

regalosPosibles([agenteBolsa], X, Y)
X=[r("paraguas",30),r("reloj",50)], Y=80
X=[r("cena",100)], Y=100

regalosPosibles([poliza, seguro, cuenta],X, Y)
X=[r("paraguas",30),r("reloj",50),r("spa",200)], Y=280
X=[r("paraguas",30),r("reloj",50),r("cena",100)], Y=180
X=[r("paraguas",30),r("spa",200)], Y=230
X=[r("reloj",50),r("spa",200)], Y=250
X=[r("viaje",300)], Y=300
X=[r("spa",200),r("cena",100)], Y=300
X=[r("spa",200)], Y=200
```

Ejercicio 14

En una software factory se utiliza un software para cargar los bugs que se detectan en el ambiente de pruebas y que el ambiente de desarrollo debe solucionar en una próxima versión. Se cuenta con una base de conocimiento con hechos del tipo `personal(Nombres)` que permite establecer los `Nombres` del personal registrado en el ámbito de desarrollo; `desarrollador(Nombre, Categorias, CodigoNivel)` que relacionan a cada desarrollador con su `Nombre`, las `Categorías` de bugs que puede reparar, y el `CodigoNivel` que le corresponde; y `niveles(Niveles)` que permite establecer la lista de `Niveles` de trabajo, donde cada elemento es un functor `n(codigoNivel, severidad)`, que determina para un `codigoNivel`, cuál es la `severidad` máxima de los bugs que puede reparar.

Se debe completar el programa Prolog que permita realizar consultas del tipo `asignarBugs(Bugs, Asignaciones)` tal que a partir de una lista de `Bugs`, compuesta por functores del tipo `b(id, categoria, severidad)`, se obtengan todas las posibles combinaciones de `Asignaciones`, como una lista de functores del tipo `a(id, nombre)`, que indica para cada `id` de bug, el `nombre` del desarrollador que puede repararlo. Se debe realizar la asignación teniendo en cuenta que un desarrollador sólo puede reparar bugs que pertenezcan a una de sus categorías, y cuya severidad sea menor o igual a la máxima correspondiente a su nivel.

Se cuenta con:

```
personal([jorge, mirta, sofia, laura, pedro, juan]).
desarrollador(mirta, [c1,c2], junior).
desarrollador(laura, [c2,c3,c1], senior).
desarrollador(pedro, [c2,c3,c4], semisenior).
desarrollador(jorge, [c1,c4], senior).
desarrollador(sofia, [c2,c4], semisenior).
desarrollador(juan, [c3,c1], junior).
niveles([n(junior, 3), n(semisenior, 7), n(senior, 10)]).
```

Ejemplos:

```
asignarBugs([b(b1, c2, 5), b(b2, c4, 8), b(b3, c4, 2)], AS)
AS = [a("b1","sofia"),a("b2","jorge"),a("b3","pedro")]
AS = [a("b1","laura"),a("b2","jorge"),a("b3","sofia")]
AS = [a("b1","laura"),a("b2","jorge"),a("b3","pedro")]
AS = [a("b1","pedro"),a("b2","jorge"),a("b3","sofia")]


asignarBugs([b(b1, c1, 2), b(b2, c3, 8), b(b3, c4, 8)], AS)
AS = [a("b1","mirta"),a("b2","laura"),a("b3","jorge")]
AS = [a("b1","juan"),a("b2","laura"),a("b3","jorge")]

asignarBugs([b(b1, c1, 9), b(b2, c4, 9), b(b3, c2, 9)], AS)
False
```

Ejercicio 15

Un grupo de amigos van muy seguido al cine, y necesitan ayuda para saber qué película elegir.

En la base de conocimiento se vuelca la información a través de hechos del tipo `datosDePeliculas(ListaDePeliculas)` que permite establecer la `ListaDePeliculas` que están en cartelera, donde cada uno de los elementos de la lista es un functor del tipo `p(titulo, genero, actores)`, que relacionan el titulo de una película con su género y los actores principales de la misma; y `preferencias(Persona, Generos, ActoresQueOdia)` que indican para cada `Persona` que conforma el grupo cuáles son los `Generos` de películas que está dispuesto a ver, y los `ActoresQueOdia`.

Completar el programa Prolog que permita realizar consultas del tipo `elegir(Personas, Peliculas, PeliculasElegidas)` que determinen sobre la lista de títulos de `Peliculas` indicada, cuáles son las `PeliculasElegidas` que todas las `Personas` del conjunto podrían ver, teniendo en cuenta sólo aquellas que pertenecen a un género que gusta a todos, y que no cuenten con ningún actor "odiado" por alguno de los integrantes del grupo.

Se cuenta con:

```
datosDePeliculas([p("Arma mortal", "Accion", ["Danny Glover", "Mel Gibson"]),
        p("Atraccion fatal", "Suspenso", ["Michael Douglas", "Glen Close"]),
        p("Cadena de favores", "Drama", ["Kevin Spacey", "Helen Hunt"]),
        p("Cuando Harry conocio a Sally", "Romantica", ["Meg Ryan", "Billy Cristal"]),
        p("El cliente", "Suspenso", ["Susan Sarandon", "Tommy Lee Jones"]),
        p("La sociedad de los poetas muertos", "Drama", ["Robin Williams", "Ethan Hawke"]),
        p("Mejor imposible", "Comedia", ["Jack Nicholson", "Helen Hunt"]),
        p("Papa por siempre", "Comedia", ["Robin Williams", "Sally Field"]),
        p("Sintonia de amor", "Romantica", ["Meg Ryan", "Tom Hanks"]),
        p("Tienes un email", "Romantica", ["Tom Hanks", "Meg Ryan"]),
        p("Buenos muchachos", "Drama", ["Robert De Niro", "Joe Pesci"]) ]).

preferencias(julian,["Accion", "Comedia", "Suspenso"], ["Meg Ryan", "Billy Cristal", "Jack Nicholson"]).
preferencias(pedro, ["Comedia", "Drama", "Romantica"], ["Joe Pesci", "Helen Hunt"]).
preferencias(lucia, ["Suspenso", "Drama", "Comedia"], ["Glenn Close", "Tommy Lee Jones"]).
```

Ejemplos:

```
elegir([julian, lucia], ["Tienes un email","El cliente", "Cadena de favores"], X)
X=[]

elegir([lucia, pedro, julian], ["Sintonia de amor","Mejor imposible","Buenos muchachos","Papa por siempre"], X)
X=["Papa por siempre"]
```