PROLOG

PARTE I: BASE DE CONOCIMIENTO, HECHOS Y REGLAS

1. Considere los siguientes hechos y reglas:

```
predecesor(luisa, veronica).
predecesor(guillermo, veronica).
predecesor(guillermo, lucia).
predecesor(veronica, maria).
predecesor(veronica, ruben).
predecesor(ruben, rafael).
predecesor(ruben, carolina).
anterior(X,Y):- predecesor(X,Y).
anterior(X,Y):- predecesor(X,Z), anterior(Z,Y).
hermanos(X,Y):- predecesor(Z,X), predecesor(Z,Y).
```

En base a sus conocimientos, señale el comportamiento y la respuesta, que arrojaría el motor de inferencias de prolog, si se realizan las siguientes consultas (Suponer que se solicita todas por *backtracking*):

a) `?predecesor(guillermo, carla).`
b) `?predecesor(guillermo, X).`
c) `?predecesor(Y, veronica).`
d) `?predecesor(ruben, carolina).`
e) `?predecesor(Y, X).`
f) `?anterior(verónica, X).`
g) `?anterior(X, rafael).`
h) `?hermanos(veronica,X).`

2. Dada las siguientes clausulas:

```
n(0).
n(s(X)) :- n(X).
m(0,s(Y)) :- n(Y).
m(s(X),s(Y)) :- m(X,Y).
```

Dibujar los árboles de resolución correspondiente al programa y a las siguientes preguntas, indicando las respuestas obtenidas.

- `?- m(X,s(s(0))).`
- `?- m(X,s(X)).`

PARTE II: LISTAS

3. Implemente predicados en lenguaje Prolog que permita definir operaciones básicas de listas:

```
head
tail
init
last
reverse
append
```

4. Dada una lista y un elemento, verificar si este pertenece a dicha lista.

```
pertenece (E, LISTA).
```

5. `SumaAnt(Xs,Ys)`:- Si Xs = $[x_1,x_2,\ldots,x_n]$, entonces Ys = $[y_1,y_2,\ldots,y_n]$, de tal forma que $y_1 = x_1 \square \sum_{i=1}^{n} x_i \;\; para\ i = 1..n$.

```
¿ sumaAnt ([3,2,1],Xs).
Xs = [9,8,7].
```

6. Dada una lista de enteros, represente su equivalente en notación decimal.

```
? rep ( [1,22,4], V).
V = 1224
```

7. Permitir la conversión entre mayúscula a minúscula.

```
? may_min([“LenguaJes DE ProGramación”],V)
V = [“lenguajes de programación”].
```

8. `subeYbaja (Xs):-` se cumple si Xs tiene forma ascendente y descendente, (creciente hasta un elemento y decreciente desde el mismo hasta el final de la lista).

```
? subeYbaja ([1,2,3,4,3,2,1]).
yes
```

9. `descomCambio (X,Xs)` :- Xs representa una combinación ordenada de un conjunto de enteros, (1,5,25,100), que son necesarias para alcanzar el valor exacto de X.

```
? descomCambio ([155],V).
V = [100,25,25,5].
```

10. Dada una lista, verificar si los elementos de la misma, son iguales.

11. Definir el predicado `SUBCONJUNTO(C1, C2)` que devuelva '`YES`' si `C1` es un subconjunto de `C2` y '`NO`' en caso contrario. Por ejemplo:

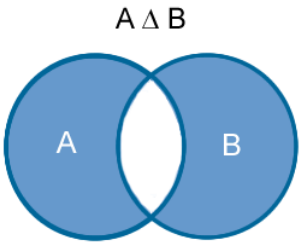
```
SUBCONJUNTO((A B),(B A)).
YES

SUBCONJUNTO((A B),((B) A)).
NO
```

12. Se desea que usted realice con el Lenguaje de Programación Prolog, un programa que dado dos conjuntos; representados cada uno como una lista, devuelva la diferencia simétrica entre ellos. La diferencia simétrica en conjuntos está definida matemáticamente como:

$A \Delta B = (A-B) \cup (B-A) = (A \cup B) - (A \cap B)$

El siguiente gráfico ilustra esta definición:



En conjuntos: A= {2,6,4,8,7,10} y B= {9,3,7,11} entonces

$A \triangle B = \{2, 6, 4, 8, 10, 9, 3, 11\}$

Llamando a la función principal de su programa:

```
? difSime ([2, 6, 4, 8, 7, 10], [9, 3, 7, 11], DS).
DS =  [2, 6, 4, 8, 10, 9, 3, 11]
```

13. Dada dos listas determinar si la primera lista es sublista de la segunda.

```
Ejemplo:
    subLista([3,2],[1,6,2,3]).
    No
    subLista([3,2],[1,6,3,2]).
    Yes
```

14. Dada una lista y una posición, eliminar los N primeros elementos de una lista y devuelve el resto.

```
Ejemplo:
        elimN([1,3,5,1,2],2,L).
        L=[5,1,2]
```

15. Dada una lista, una posición y un elemento, insertar el elemento en la posición N de la Lista.

```
Ejemplo:
        insert([1,2,3,4],1,5,R).
        R=[5,2,3,4]
```

16. Dada un elemento y una lista, armar una lista con todos los elementos menores que el.

```
Ejemplo:
        menores(3,[2,3,1,7,0],L).
        L=[2,1,0]
```

17. Dada dos listas, devolver los electos comunes entre ellas

```
Ejemplo:
        comunes([1,6,3],[2,1,8,3],L).
        L=[1,3]
```

18. Dada una matriz, rotar la matriz (en dirección de las agujas del reloj).

```
  1 2 3              7 4 1
  4 5 6      =>      8 5 2
  7 8 9              9 6 3
```

```
Ejemplo: rotar([[1,2,3],[4,5,6],[7,8,9]],M).
          M = [[7,4,1],[8,5,2],[9,6,3]]
```

19. Dada una lista, eliminar de esta todas las repeticiones.

```
eliminar(Lista)
```

20. Dada dos lista de igual tamaño, permita realizar, elemento por elemento, operaciones aritméticas básicas.

```
arit(CHAR,LISTA,LISTA,LISTA)
```

21. Dado un valor entero, trasformar su equivalente en base binaria o viceversa. `decBin(D,B):-` donde B es el valor Binario de D.

```
? decBin(7,V)

V = [1,1,1]

? decBin(V,[1,1,1])

V = 7
```

22. Sea la siguiente definición, `palabras (Cad, Pals)`, donde `Cad`, es una cadena de caracteres y `Pals` es una lista de estructuras de la forma: `c(P,N)`, donde `P` es una palabra que existe en `Cad` y `N` es el número de veces que se repite dicha palabra en la cadena. Se deben ignorar las diferencias entre mayúscula y minúscula y los caracteres de puntuación.

```
? palabras (“La ociosidad, es la madre de la filosofía.”, V)
V = [c(la,3), c(ociosidad,1), c(es,1), c(madre,1), c(de,1), c(filosofía,1)]
```

23. Construir un programa de gestión de una agencia matrimonial. El programa debe almacenar datos de las personas, así como sus preferencias y, buscar las mejores combinaciones de parejas según su criterio. (Use E/S y Base de Datos).

PARTE III ÁRBOLES BINARIOS.

24. Defina a programa que permita determinar cuando, si un árbol es isomorfo a otro. `<isIsomorfo(Arbol1, Arbol2)>.`

25. Defina una cláusula, que permita recorrer un árbol en, post-orden, pre-orden e in-orden.

```
pre_orden(Arb, Pre), post_orden(Arb, Post), in_orden(Arb, In).
```

Donde *`Pre, Post`* e *`In`*, es el recorrido pre_orden, post-orden e in-orden respectivamente sobre el árbol binario *Arb*.

26. Defina un programa, que permita sustituir un término en un árbol. *`sustituir`*`(X,Y, AbolX, ArbolY),` donde el árbol binario *`arbolY`* es el resultado de templar todas las ocurrencias de *`X`* en el árbol binario *`arbolX`* por *`Y.`*