PRACTICO NÚMERO 4 – PROGRAMACIÓN DECLARATIVA

VENTURA GINO

EJERCICIO 1:

Definir la relación resto (L1, L2) donde se verifique si L2 es la lista obtenida de L1, eliminando el primer elemento.

Pruebas:

resto([a,b,c],L).

L = [b,c]

CÓDIGO:

```
resto([_|L], L).
```

PRUEBA:

```
|
|    resto([a,b,c],L).
L = [b, c].
```

EJERCICIO 2:

Definir la relación de pertenencia(X,L) donde se verifique si X es un elemento de la lista L.

Pruebas:

pertenencia(b,[a,b,c]) = true

pertenencia(d,[a,b,c]) = false

CÓDIGO:

```
pertenencia(X,[X|_]).

pertenencia(X,[_|L]) :- pertenencia(X,L).
```

PRUEBA:

```
?- pertenencia(b,[a,b,c]).
true .

?- pertenencia(d,[a,b,c]).
false.
```

EJERCICIO 3:

Determinar el tamaño de una lista L.

Pruebas:

size([1,2,3],L).

L=3

Size([],L).

L=0

CÓDIGO:

```
tamanio([],0).

tamanio([_|L],N) :- tamanio(L,N1), N is N1+1.
```

PRUEBA:

```
?- tamanio([1,2,3],L).
L = 3.

?- tamanio([],L).
L = 0.
```

EJERCICIO 4:

Definir la relación iguales(L), que valide si todos los valores de la lista L son iguales.

Pruebas:

iguales([a,a,a]) = true

iguales[a,b,c]) = false

iguales([]) = true

CÓDIGO:

```
iguales([]).

iguales([_]).

iguales([X,X|L]) :- iguales([X|L]).
```

PRUEBA:

```
?- iguales([a,a,a]).
true .

?- iguales([a,b,c]).
false.

?- iguales([]).
true.
```

EJERCICIO 5:

Definir la relación mayor(X, Y, Z) que valide en Z cual es el mayor entre X e Y.

Pruebas:

mayor(4, 5, Z).

Z=5

mayor(7, 2, Z).

Z=7

CÓDIGO:

```
mayor(X,Y,X) :- X>Y.

mayor(X,Y,Y) :- X<Y.
```

```
?- mayor(4,5,Z).
Z = 5.

?- mayor(7,2,Z).
Z = 7 .
```

EJERCICIO 6:

Definir la relación sumLista(L,X) donde X sea la suma de los elementos de la lista L.

Pruebas:

sumLista([1,2,3],X).

X=6

CÓDIGO:

```
sumLista([],0).

sumLista([X|L],Y) :- sumLista(L,Y1), Y is X+Y1.
```

PRUEBA:

```
?- sumLista([1,2,3],X).
X = 6.
```

EJERCICIO 7:

Definir la relación orden(L) que verifique si la lista L esta ordenada en forma ascendente.

Pruebas:

orden([1,3,4,6]).

True

orden([1,6,5,2]).

False

CÓDIGO:

```
orden([_]).
orden([X,Y|L]) :- X=<Y, orden([Y|L]).
```

PRUEBA:

```
?- orden([1,3,4,6]).
true .

?- orden([1,6,5,2]).
false.
```

EJERCICIO 8:

Definir la relación listaNum(N,M,L) que devuelva L como la lista de valores desde N hasta M

Pruebas:

listaNum(3,6,L).

L = [3,4,5,6]

listaNum(3,1,L) = false

CÓDIGO:

```
lista_num(N,N,[N]).
lista_num(N,M,[N|L]) :- N<M, N1 is N+1, lista_num(N1, M, L).
```

PRUEBA:

```
?- lista_num(3,6,L).
L = [3, 4, 5, 6] .

?- lista_num(3,1,L).
false.
```

EJERCICIO 9:

Definir una función que inserte un valor en una lista ordenada en forma ascendente de tal forma que el valor quede en el lugar que le corresponda.

Pruebas:

insert(1,[],L).

L=[1]

insert(3,[1,2,4,5],L).

L=[1,2,3,4,5]

CÓDIGO:

```
insert(X,[],[X]).

insert(X,[Y|Ys], [X,Y|Ys]) :- X<Y.

insert(X, [Y|Ys], [Y|Zs]) :- X>=Y, insert(X,Ys,Zs).
```

PRUEBA:

```
?- insert(1,[],L).
L = [1] .

?- insert(3,[1,2,4,5],L).
L = [1, 2, 3, 4, 5] .
```

EJERCICIO 10:

Definir una función que indique si los valores ingresados forman un capicúa.

Pruebas:

capicua([o,s,o]) = True

capicua([1,2,1]) = True

capicua([1,2,3]) = false

CÓDIGO:

```
capicua(L) :- reverse(L,L).
```

PRUEBA:

```
?- capicua([o,s,o]).
true.

?- capicua([1,2,3]).
false.
```