LENGUAJES, TECNOLOGÍAS Y PARADIGMAS
DE PROGRAMACIÓN

TEMA 4:
PROGRAMACIÓN LÓGICA
(SOLUCIONES EJERCICIOS DE AULA)

SOLUCIONES

Tema 4: Ejercicios de Aula

---

1. B
2. A
3. B
4. A
5. D
6. B
7. B
8. C
9. B
10. D
11. C
12. A
13. D
14. B
15. C
16. C
17. B

SOLUCIONES

Ejercicios de Aula: Cuestiones

1.
```
hombre(alfredo).
hombre(felipe).
hombre(francisco).
mujer(sonia).
mujer(eva).
mujer(carmen).
bebe(alfredo, whisky).
bebe(alfredo, ron_cola).
bebe(felipe, cerveza).
bebe(felipe, gin_tonic).
bebe(felipe,ron_cola).
bebe(francisco, vino).
bebe(francisco, malibu).
bebe(sonia, gin_tonic).
bebe(sonia, malibu).
bebe(eva, vino).
bebe(eva, cerveza).
bebe(carmen, whisky).
bebe(carmen, ron_cola).

pareja(X,Y):-hombre(X), mujer(Y), bebe(X,Z), bebe(Y,Z).

pareja2(X,Y):-hombre(X), mujer(Y), bebe(X,Z), bebe(Y,Z),
              bebe(X,W), bebe(Y,W), Z \== W.
```

2.
```
bebe(pepe,X):-bebe(alfredo,X).
bebe(elena,X):-bebe(sonia,X);bebe(felipe,X).
```

3.
```
vecinas(almeria,granada).
vecinas(jaen,granada).
vecinas(cordoba,granada).
vecinas(malaga,granada).
vecinas(malaga,sevilla).
vecinas(malaga,cordoba).
vecinas(malaga,cadiz).
vecinas(huelva,cadiz).
vecinas(sevilla,cadiz).

fronterizas(X,Y):-vecinas(X,Y);vecinas(Y,X).

viaje(X,Y):-fronterizas(X,Y).
viaje(X,Y):-fronterizas(X,Z),fronterizas(Z,Y).
```

4.
```
padece(pedro,gripe).
padece(pedro,hepatitis).
padece(juan,hepatitis).
padece(maria,gripe).
padece(carlos,intoxicacion).

sintoma(fiebre,gripe).
sintoma(cansancio,hepatitis).
sintoma(vomito,intoxicacion).
sintoma(cansancio,gripe).

suprime(aspirina,fiebre).
suprime('Motilium',vomito).

alivia(X,Y):-sintoma(Z,Y),suprime(X,Z).

toma_farmaco(P,F):-padece(P,E),alivia(F,E).
```

   - gripe, hepatitis

     ```
     ?- padece(pedro,X).
     X = gripe ;
     X = hepatitis.
     ```

   - gripe

     ```
     ?- padece(maria,X).
     X = gripe.
     ```

   - pedro, maria

     ```
     ?- padece(X,gripe).
     X = pedro ;
     X = maria ;
     false.
     ```

   - fiebre, cansancio

     ```
     ?- padece(pedro,E),sintoma(S,E).
     E = gripe,
     S = fiebre ;
     E = gripe,
     S = cansancio ;
     E = hepatitis,
     S = cansancio ;
     false.
     ```

- carlos

  ```
  ?- padece(P,E),sintoma(vomito,E).
  P = carlos,
  E = intoxicacion.
  ```

- pedro, juan, maria

  ```
  ?- padece(P,E),sintoma(cansancio,E).
  P = pedro,
  E = gripe ;
  P = pedro,
  E = hepatitis ;
  P = juan,
  E = hepatitis ;
  P = maria,
  E = gripe ;
  false.
  ```

- aspirina

  ```
  ?- toma_farmaco(pedro,F).
  F = aspirina ;
  false.
  ```

- no

  ```
  ?- toma_farmaco(juan,F), toma_farmaco(maria,F).
  false.
  ```

5. ```
   rufian(bertoldo).
   rufian(bartolo).

   noble(romeo).
   noble(bertoldo).

   plebeyo(bartolo).

   dama(gertrudis).
   dama(julieta).

   hermosa(julieta).

   desea(X,Y):-plebeyo(X), dama(Y).
   desea(X,Y):-noble(X), dama(Y), hermosa(Y).

   rapta(X,Y):-rufian(X), desea(X,Y).
   ```
   ```

- bertoldo

  ```
  ?- noble(X),rufian(X).
  X = bertoldo.
  ```

- a nadie

  ```
  ?- rapta(romeo,X).
  false.
  ```

- bertoldo y bartolo

  ```
  ?- rapta(X,julieta).
  X = bertoldo ;
  X = bartolo ;
  false.
  ```

- bertoldo a julieta y bartolo a gertrudis y julieta

  ```
  ?- rapta(X,Y).
  X = bertoldo,
  Y = julieta ;
  X = bartolo,
  Y = gertrudis ;
  X = bartolo,
  Y = julieta ;
  false.
  ```

- a gertrudis y julieta

  ```
  ?- desea(bartolo,X).
  X = gertrudis ;
  X = julieta ;
  false.
  ```

- a julieta

  ```
  ?- desea(romeo,X).
  X = julieta.
  ```

- a julieta

  ```
  ?- dama(X), hermosa(X), desea(bartolo,X).
  X = julieta ;
  false.
  ```

6. ```
   palindromo(L):-reverse(L,L).

   /*Alternativa recursiva
   palindromo([]).
   palindromo([X]).
   palindromo([X|L]):-append(T,[X],L), palindromo(T).
   */
   ```
   ```

7. ```
   vertical(segmento(punto(X,_),punto(X,_))).
   horizontal(segmento(punto(_,Y),punto(_,Y))).
   ```

8. ```
   f(X,0):-X<3.
   f(X,2):- 3=<X, X<6.
   f(X,4):-6=<X.
   ```

9. Comprueba si son iguales entre sí todos los elementos de una lista.

10. ```
    longitud_impar([_]).
    longitud_impar([_,_|L]):-longitud_impar(L).
    ```

11. ```
    longitud_par([]).
    longitud_par([_,_|L]):-longitud_par(L).
    ```

12. ```
    longitud_impar([_]).
    longitud_impar([_|L]):-longitud_par(L).

    longitud_par([]).
    longitud_par([_|L]):-longitud_impar(L).
    ```

13. ```
    remove(X,[],[]).
    remove(X,[X|L],Z):-remove(X,L,Z).
    remove(X,[Y|L],[Y|Z]):-X\=Y, remove(X,L,Z).

    remove_rep([],[]).
    remove_rep([X],[X]).
    remove_rep([X|L],[X|Z]):-member(X,L), remove(X,L,M),remove_rep(M,Z).
    remove_rep([X|L],[X|Z]):-not(member(X,L)), remove_rep(L,Z).
    ```

14. ```
    repeat(_,[],0).
    repeat(X,[X|L],N):-repeat(X,L,M), N is M+1.
    repeat(X,[Y|L],N):-X\=Y, repeat(X,L,N).
    ```

15. ```
    largest([Y],Y).
    largest([X,Y|L],Z):-X>=Y,largest([X|L],Z).
    largest([X,Y|L],Z):-X<Y,largest([Y|L],Z).
    ```

16. ```
    split(U,[],[],[]).
    split(U,[X|L],[X|Z],V):- U>=X, split(U,L,Z,V).
    split(U,[X|L],Z,[X|V]):- U<X, split(U,L,Z,V).
    ```

17. ```
    add([],0).
    add([X|L],A):- add(L,B), A is X+B.
    ```