

**Problema A:** [*veins.pl*]

Donades cinc persones que tenen cinc cases de colors diferents, i cinc professions, animals, begudes i nacionalitats diferents, i sabent que:

- 1 - El que viu a la casa vermella és del Perú
- 2 - Al francès li agrada el gos
- 3 - El pintor és japonès
- 4 - Al xinès li agrada el rom
- 5 - L'hongarès viu en la primera casa
- 6 - Al de la casa verda li agrada el conyac
- 7 - La casa verda està just a l'esquerra de la blanca
- 8 - L'escultor cria caragols
- 9 - El de la casa groga és actor
- 10 - El de la tercera casa beu cava
- 11 - El que viu al costat de l'actor té un cavall
- 12 - L'hongarès viu al costat de la casa blava
- 13 - Al notari l'agrada el whisky
- 14 - El que viu al costat del metge té un esquiol
- 15 - El que té un gat no és a qui agrada la cervesa

escriu un programa Prolog que averigüe per a cada persona totes les seves característiques de la forma `[numcasa,color,professió,animal,beguda,pais]` averiguables.

Nota: partint d'una solució `[ [1,A1,B1,C1,D1,E1], ..., [5,A5,B5,C5,D5,E5] ]`, es poden imposar totes les condicions sobre aquesta amb `member` o similars.

**Problema B:** [*aigües.pl*, *missioners.pl*, *cavall.pl*]

Adapta (és obligatori) l'esquema Prolog que els segueix per a resoldre els tres problemes que veieu a continuació. Pensa com representar els estats, i quins passos entre estats n'hi ha. Per exemple, en el problema de les galledes d'aigua, l'estat inicial pot ser `[0,0]`, i el final `[0,4]`. En tots tres problemes el cost de cada transició/pas és 1.

**B.1. Aigües:** disposem d'una aixeta d'aigua, una galleda de 5 litres i un altre de 8 litres. Es pot abocar el contingut d'una galleda en un altre (fins a buidar el primer, o fins a omplir l'altre), omplir una galleda, o buidar una galleda del tot. Escriure un programa Prolog que digui la seqüència més curta d'operacions per a obtenir exactament 4 litres d'aigua en la galleda de 8 litres.

**B.2. Missioners i caníbals:** busquem la manera més ràpida per tal que tres missioners i tres caníbals travessin un riu en una canoa que pot ser utilitzada per 1 o 2 persones (missioners o caníbals), però sempre evitant que els missioners quedin en minoria en qualsevol riba (per raons òbvies).

**B.3. Cavall:** donat un natural  $n > 0$ , una posició inicial  $(Fila_I, Columna_I)$ , una posició final  $(Fila_F, Columna_F)$ , i un nombre de passos  $P$ , trobar un camí de  $(Fila_I, Columna_I)$  a  $(Fila_F, Columna_F)$ , en un tauler d'escacs de  $n \times n$  en exactament  $P$  passos de cavall, i passant per caselles diferents. El programa ha de fallar si per a la  $n$  en qüestió no existeix tal camí. Aquí teniu un exemple d'entrada:

```
midaTauler(7).
posInicial(1,5).
posFinal(6,2).
numMoviments(6).
```

Esquema Prolog de *cami\_cos\_minim.pl*:

```
main :- EstatInicial = ...,   EstatFinal = ...,
        between(0, 1000, CostMax),           % Busquem solució de cost 0; si no, de 1, etc.
        cami(CostMax, EstatInicial, EstatFinal, [EstatInicial], Cami),
        reverse(Cami, Cami1), write(Cami1), write(' amb cost '), write(CostMax), nl, halt.

cami(0, E, E, C, C).           % Cas base: quan l'estat actual és l'estat final.
cami(CostMax, EstatActual, EstatFinal, CamiFinsAra, CamiTotal) :-
    CostMax > 0,
    unPas(CostPas, EstatActual, EstatSeguent), % En B1 i B2, CostPas és 1.
    \+ member(EstatSeguent, CamiFinsAra),
    CostMax1 is CostMax-CostPas,
    cami(CostMax1, EstatSeguent, EstatFinal, [EstatSeguent|CamiFinsAra], CamiTotal).

unPas(...) :- ...
...
```

### Problema C: [*mastermind.pl*]

*Mastermind* és un joc on un jugador (defensor) s'inventa un codi secret i l'altre jugador (atacant) ha d'esbrinar-ho. El codi és una seqüència de 4 colors a triar entre vermell (v), blau (b), groc (g), lila (l), taronja (t) i marró (m). L'atacant té un nombre finit d'intents per a trencar el codi. En cada intent, l'atacant preguntarà per una seqüència de 4 colors i el defensor respondrà amb dos números ( $E, D$ ), sent  $E$  el nombre de colors que l'atacant ha encertat en la posició Exacta, i  $D$  el nombre de colors que ha encertat però en una posició Diferent. Exemple:

El defensor tria el codi [v,t,t,b].

Intent 1: l'atacant pregunta per [v,b,g,l]. La resposta és [1,1].

El primer 1 és degut a la primera posició (v).

El segon 1 és degut a la b, que no està en la posició correcta.

Intent 2: l'atacant pregunta per [m,t,g,l]. La resposta és [1,0].

Intent 3: l'atacant pregunta per [g,l,g,l]. La resposta és [0,0].

Intent 4: l'atacant pregunta per [v,b,m,m]. La resposta és [1,1].

Intent 5: l'atacant pregunta per [v,t,b,t]. La resposta és [2,2].

Intent 6: l'atacant pregunta per [v,t,t,b]. La resposta és [4,0]. L'atacant guanya el joc.

**C.1.** Construeix un predicat `resposta(Codi, Intent, E, D)` que, donat un Codi i un Intent calculi els números  $E, D$  de la resposta. Exemples:

```
?- resposta([v,t,t,b], [m,t,g,l], E, D).
```

```
E = 1,
```

```
D = 0.
```

```
?- resposta([v,t,t,b], [v,t,b,t], E, D).
```

```
E = D, D = 2.
```

```
?- resposta([v,b,g,l], [v,t,b,t], E, D).
```

```
E = D, D = 1.
```

Consell: calcular  $E$  hauria de ser fàcil. Per calcular  $D$ , pots calcular  $T$  (nombre de posicions a l'intent que encerta un color correcte, en la posició exacta o en una posició diferent) i usar el fet que  $D = T - E$ .

**C.2.** Volem ara ajudar l'atacant a guanyar el joc, suggerint-li intents. Assumeix que ens donen una clàusula de la forma:

```
% intents(L): L és una llista de parelles [Intent, Resposta]
intents([ [ [v,b,g,1], [1,1] ], [ [m,t,g,1], [1,0] ], [ [g,1,g,1], [0,0] ],
         [ [v,b,m,m], [1,1] ], [ [v,t,b,t], [2,2] ] ]).
```

que representa l'històric dels intents fets fins ara per l'atacant. Construeix un nou predicat `nouIntent(A)` que genera un nou intent `A` tal que, si `A` fora el codi a descobrir, llavors tots els intents en l'històric tindrien com a resposta la que justament vam obtenir. O, dit d'un altra manera, el nou intent `A` només podrà ser el codi secret si és *consistent* amb el que trobem a l'històric. Exemple:

```
?- nouIntent(A).
A = [v, t, t, b]
```

Pots provar algun *Mastermind* en línia i utilitzar aquest predicat, possiblement adaptant els colors, per a guanyar en un nombre raonable de passos.