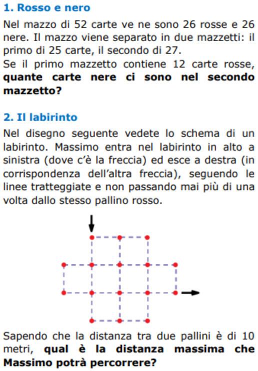
LAAI module 1

Gian Marco Baroncini

What have I done?

I have made **3** exercises in **prolog**.

The first two from "ALLENAMENTI GIOCHI D'AUTUNNO 2021". https://giochimatematici.unibocconi.it/images/autunno/2021/practiceq.pdf



1. Rosso e nero

Nel mazzo di 52 carte ve ne sono 26 rosse e 26 nere. Il mazzo viene separato in due mazzetti: il primo di 25 carte, il secondo di 27.
Se il primo mazzetto contiene 12 carte rosse, **quante carte nere ci sono nel secondo mazzetto?**

2. Il labirinto

Nel disegno seguente vedete lo schema di un labirinto. Massimo entra nel labirinto in alto a sinistra (dove c'è la freccia) ed esce a destra (in corrispondenza dell'altra freccia), seguendo le linee tratteggiate e non passando mai più di una volta dallo stesso pallino rosso.

Sapendo che la distanza tra due pallini è di 10 metri, **qual è la distanza massima che Massimo potrà percorrere?**

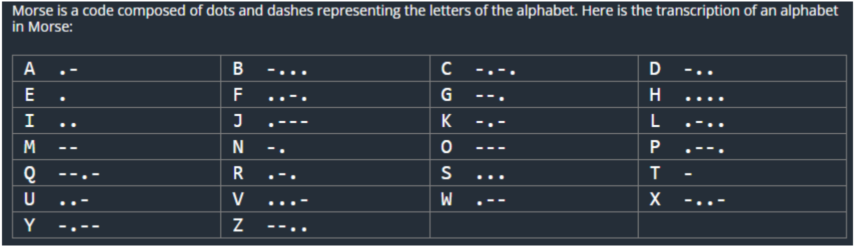
Solutions here: https://giochimatematici.unibocconi.it/images/autunno/2021/practicea.pdf

What have I done?

The third exercise is **"The Resistance"** from `https://www.codingame.com/ide/puzzle/the-resistance`.

"You work in the National Resistance Museum and you just uncovered hundreds of documents which contain coded Morse transmissions. In the documents, none of the spaces have been transcribed to separate the letters and the words hidden behind the Morse sequence. Therefore, there may be several interpretations of any single decoded sequence. Your program must be able to determine the number of different messages that it's possible to obtain from one Morse sequence and a given dictionary."...

Description



Morse is a code composed of dots and dashes representing the letters of the alphabet. Here is the transcription of an alphabet in Morse:

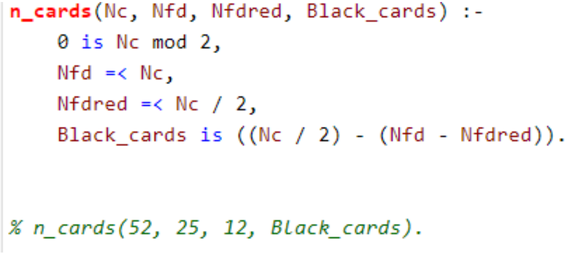
A .-	B -...	C -.-.	D -..
E .	F ..-.	G --.	H ....
I ..	J .---	K -.-	L .-..
M --	N -.	O ---	P .--.
Q --.-	R .-.	S ...	T -
U ..-	V ...-	W .--	X -..-
Y -.--	Z --..		

Since none of the spaces have been transcribed, there may be several possible interpretations. For example, the sequence -....–.-. could be any of the following: BAC, BANN, DUC, DU TETE, ...

First Exercise

Simple exercise, just to get familiar with the language.

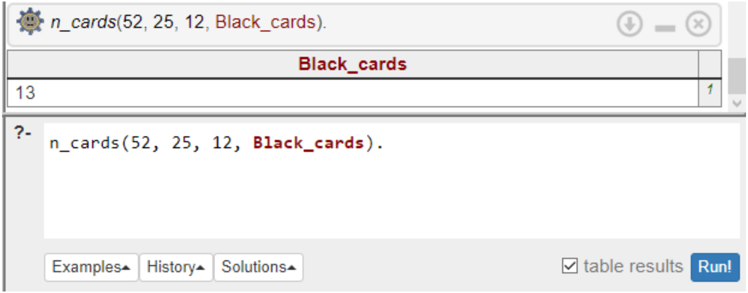
Code:



```
n_cards(Nc, Nfd, Nfdred, Black_cards) :-
    0 is Nc mod 2,
    Nfd =< Nc,
    Nfdred =< Nc / 2,
    Black_cards is ((Nc / 2) - (Nfd - Nfdred)).


% n_cards(52, 25, 12, Black_cards).
```

Result



```
n_cards(52, 25, 12, Black_cards).
```

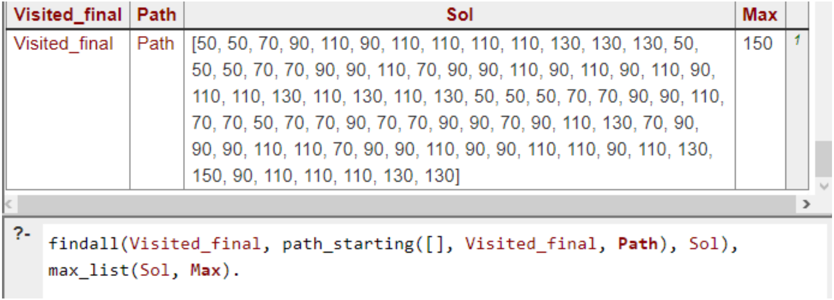
Black_cards	
13	1

```
?- n_cards(52, 25, 12, Black_cards).
```

Second exercise - Implementation choices:

- Coding the maze connections with: **link(node1, node2)**.
- Rule **path_starting** in order to start from node 1.
- Rule **sub_path** which looks for a link in both directions link(X,Y) and link(Y,X). Furthermore the base case which check if the second node is 13.

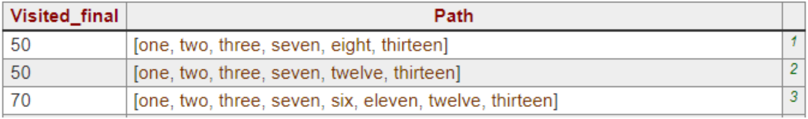
Queries and Results



Visited_final	Path	Sol	Max	
Visited_final	Path	[50, 50, 70, 90, 110, 90, 110, 110, 110, 110, 130, 130, 130, 50, 50, 50, 70, 70, 90, 90, 110, 70, 90, 90, 110, 90, 110, 90, 110, 90, 110, 110, 130, 110, 130, 110, 130, 50, 50, 50, 70, 70, 90, 90, 110, 70, 70, 50, 70, 70, 90, 70, 70, 90, 90, 70, 90, 110, 130, 70, 90, 90, 90, 110, 110, 70, 90, 90, 110, 90, 90, 110, 110, 90, 110, 130, 150, 90, 110, 110, 110, 130, 130]	150	1

```
?- findall(Visited_final, path_starting([], Visited_final, Path), Sol),
   max_list(Sol, Max).
```

Alternatively to see the various constructions of the solutions the query is: **path_starting([], Visited_final, Path).** The last one prints solution by solution with path. Example:



Visited_final	Path	
50	[one, two, three, seven, eight, thirteen]	1
50	[one, two, three, seven, twelve, thirteen]	2
70	[one, two, three, seven, six, eleven, twelve, thirteen]	3

Third exercise

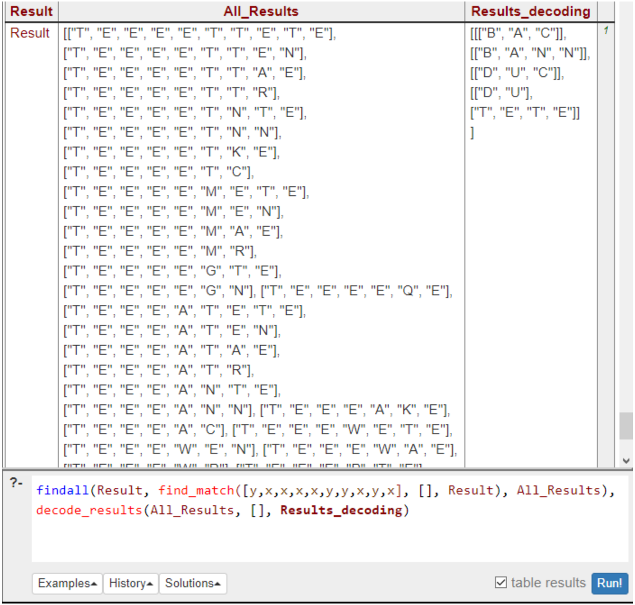
- Similar approach to the previous one for transcribing **knowledge**.

- "." is encoded as x, "-" as y.

- Search for a two-stage solution, such as "encoder-decoder" architecture. First is looking for all possible matches, second looks for corresponding sequence-dictionary.

- Respectively rule **find_match** and rule **decode_results**. The latter iterates sequence by sequence and exploits **decode_single_results** to find matches taking into account letter by letter of each sequence.

Queries and Results



Result	All_Results	Results_decoding	
Result	[["T", "E", "E", "E", "E", "T", "T", "E", "T", "E"], ["T", "E", "E", "E", "E", "T", "T", "E", "N"], ["T", "E", "E", "E", "E", "T", "T", "A", "E"], ["T", "E", "E", "E", "E", "T", "T", "R"], ["T", "E", "E", "E", "E", "T", "N", "T", "E"], ["T", "E", "E", "E", "E", "T", "N", "N"], ["T", "E", "E", "E", "E", "T", "K", "E"], ["T", "E", "E", "E", "E", "T", "C"], ["T", "E", "E", "E", "E", "M", "E", "T", "E"], ["T", "E", "E", "E", "E", "M", "E", "N"], ["T", "E", "E", "E", "E", "M", "A", "E"], ["T", "E", "E", "E", "E", "M", "R"], ["T", "E", "E", "E", "E", "G", "T", "E"], ["T", "E", "E", "E", "E", "G", "N"], ["T", "E", "E", "E", "E", "Q", "E"], ["T", "E", "E", "E", "A", "T", "E", "T", "E"], ["T", "E", "E", "E", "A", "T", "E", "N"], ["T", "E", "E", "E", "A", "T", "A", "E"], ["T", "E", "E", "E", "A", "T", "R"], ["T", "E", "E", "E", "A", "N", "T", "E"], ["T", "E", "E", "E", "A", "N", "N"], ["T", "E", "E", "E", "A", "K", "E"], ["T", "E", "E", "E", "A", "C"], ["T", "E", "E", "E", "W", "E", "T", "E"], ["T", "E", "E", "E", "W", "E", "N"], ["T", "E", "E", "E", "W", "A", "E"], ["T", "E", "E", "E", "W", "R"], ["T", "E", "E", "E", "R", "T", "E"]	[[["B", "A", "C"]], [["B", "A", "N", "N"]], [["D", "U", "C"]], [["D", "U"], ["T", "E", "T", "E"]]]	1

```
?- findall(Result, find_match([y,x,x,x,x,y,y,x,y,x], [], Result), All_Results),
decode_results(All_Results, [], Results_decoding)
```