Informação

Destacar pergunta

Consider the following knowledge base regarding books and authors.

PT

```
%author(AuthorID, Name, YearOfBirth, CountryOfBirth).
author(1,       'John Grisham',         1955,   'USA').
author(2,       'Wilbur Smith',         1933,   'Zambia').
author(3,       'Stephen King',         1947,   'USA').
author(4,       'Michael Crichton',     1942,   'USA').

%book(Title, AuthorID, YearOfRelease, Pages, Genres).
book('The Firm',                1,      1991,   432,    ['Legal thriller']).
book('The Client',              1,      1993,   422,    ['Legal thriller']).
book('The Runaway Jury',        1,      1996,   414,    ['Legal thriller']).
book('The Exchange',            1,      2023,   338,    ['Legal thriller']).

book('Carrie',                  3,      1974,   199,    ['Horror']).
book('The Shining',             3,      1977,   447,    ['Gothic novel', 'Horror', 'Psychological horror']).
book('Under the Dome',          3,      2009,   1074,   ['Science fiction', 'Political']).
book('Doctor Sleep',            3,      2013,   531,    ['Horror', 'Gothic', 'Dark fantasy']).

book('Jurassic Park',           4,      1990,   399,    ['Science fiction']).
book('Prey',                    4,      2002,   502,    ['Science fiction', 'Techno-thriller', 'Horror', 'Nanopunk']).
book('Next',                    4,      2006,   528,    ['Science fiction', 'Techno-thriller', 'Satire']).
```

Answer questions 1 to 5 **WITHOUT** using multiple solution predicates (findall, setof, and bagof) and **WITHOUT** using any SICStus library.

Pergunta 1

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***book_genre(?Title, ?Genre)***, which relates a book title to one of the genres of that book.

PT

```
book_genre(Title, Genre) :-
    book(Title, _AuthorID, _YearOfRelease, _Pages, Genres),
    member(Genre, Genres).
```

```
book_genre(Title, Genre):-
        book(Title, _AuthorID, _Year, _Pages, Genres),
        member(Genre, Genres).

% member/2 is used to retrieve only one genre; additional solutions are returned via backtracking.
```

Pergunta 2

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***author_wrote_book_at_age(?Author, ?Title, ?Age)***, which unifies an author's name (***Author***) with the ***Title*** of a book they wrote and the ***Age*** of the author at the time the book was released. Consider the age of the author as the difference between the book's year of release and the author's year of birth. Example:

PT

```
| ?- author_wrote_book_at_age('John Grisham', 'The Client', Age).
Age = 38 ?
```

```
author_wrote_book_at_age(Author, Title, Age) :-
    author(AuthorID, Author, YearOfBirth, _CountryOfBirth),
    book(Title, AuthorID, YearOfRelease, _Pages, _Genres),
    Age is YearOfRelease - YearOfBirth.
```

```
author_wrote_book_at_age(Author, Title, Age):-
        author(AuthorID, Author, YearOfBirth, _CountryOfBirth),
        book(Title, AuthorID, YearOfRelease, _Pages, _Genres),
        Age is YearOfRelease - YearOfBirth.
```

Pergunta 3

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***youngest_author(?Author)***, which unifies ***Author*** with the name of the person who became an author at the youngest age. If more than one author published their first book at the same age, different results should be provided via backtracking. Example:

PT

```
| ?- youngest_author(Author).
Author = 'Stephen King' ? ;
no
```

```
youngest_author(Author) :-
    author_wrote_book_at_age(Author, _Title, Age),
    \+((author_wrote_book_at_age(_, _Title9, Age1),Age1<Age)).
```

```
youngest_author(Author):-
        author_wrote_book_at_age(Author, _Title, _Age),
        \+ ( author_wrote_book_at_age(_Author2, _Title2, _Age2), _Age2 < _Age ).

% Negation can be used to achieve a simple implementation of this predicate.
```

Pergunta 4

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***genres(+Title)***, which, without using recursion, print all genres of the given book ***Title*** to the terminal (one per line, in the order by which they appear in the knowledge base). The predicate must always succeed. Solutions using recursion obtain at most 25% of this question's grade. Examples:

PT

```
| ?- genres('The Firm').
Legal thriller
yes
| ?- genres('The Shining').
Gothic novel
Horror
Psychological horror
yes
```

```
genres(Title) :-
    book(Title, _AuthorID, _YearOfRelease, _Pages, Genres),
    member(Genre, Genres),
    print(Genre),nl, fail.
genres(_).
```

```
genres(Book):-
        book(Book, _AuthorID, _Year, _Pages, Genres),
        member(Genre, Genres),
        write(Genre), nl,
        fail.
genres(_Book).

% This solution uses a failure driven loop (slides 19-21 of slides PL6) to print the genres one by one.
% Note the second clause to make sure the predicate always succeeds (even if the book does not exist)
```

Pergunta 5

Respondida

Pontuou 0,200 de 1,400

Destacar pergunta

Implement ***filterArgs(+Term, +Indexes, ?NewTerm)***, which receives a Prolog ***Term*** as the first argument and a list of ***Indexes*** (starting at 1; you can assume the values are positive integers, sorted ascendingly), unifying the third argument with a ***new term*** containing only the arguments in the positions indicated by the indexes in the list. Examples:

PT

```
| ?- filterArgs( book('The Firm', 1, 1991, 432, ['Legal thriller']), [1, 3, 5], NewTerm).
NewTerm = book('The Firm', 1991, ['Legal thriller']).
| ?- filterArgs( author(1, 'John Grisham', 1955, 'USA'), [2, 3], NewTerm).
NewTerm = author('John Grisham', 1955).
```

```
filterArgs(Term, Indexes, NewTerm) :-
    Term =.. NewTerm.
filterbyind([H|T1], L, [R|T2]) :-
    nth1(H, L, R),
    filterbyind(T1, L, T2).
filterbyind([],_,[]).
```

```
filterArgs(Term, Indexes, NewTerm):-
        Term =.. [F|_Args],
        filterAux(Indexes, Term, NewArgs),
        NewTerm =.. [F|NewArgs].

filterAux([], _, []).
filterAux([Idx|Idxs], Term, [Arg|Rest]):-
        arg(Idx, Term, Arg),
        filterAux(Idxs, Term, Rest).

% The univ (=..) operator is used to deconstruct the original term and to construct the resulting term.
% The arg/3 predicate is used to access the arguments in the specified indexes.
```

Informação

Destacar pergunta

In the following questions, you can use set predicates (findall/3, bagof/3 and setof/3) as well as any SICStus library.

PT

Pergunta 6

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***diverse_books(-Books)***, which returns a list of all the book titles (can be only one or more) with the greatest number of genres (if more than one exists, any order is accepted). Example:

PT

```
| ?- diverse_books(Books).
Books = ['Prey'] ? ;
no
```

```
diverse_books(Books) :-
    findall(Lengths, (book(Title, _AuthorID, _YearOfRelease, _Pages, Genres), length(Genres, Lengths)), Lengths),
    sort(Lengths, L), reverse(L, [H|_]),
    findall(Title,(book(Title, _AuthorID, _YearOfRelease, _Pages, Genres), length(Genres, H)), Books).
```

```
diverse_books(Books):-
        findall(Title, ( book(Title, _, _, _, Genres), length(Genres, Len),
                        \+ (( book(_,_,_,_,Genres2), length(Genres2, Len2), Len2>Len )) ), Books).
```

Share

Pergunta 7

Respondida

Pontuou 0,700 de 1,400

Destacar pergunta

Implement ***country_authors(?Country, ?Authors)***, which unifies ***Country*** with a country of birth from the knowledge base and ***Authors*** with the list of all authors who were born in that country. Example:

PT

```
| ?- country_authors(Country, Authors).
Country = 'USA',
Authors = ['John Grisham','Stephen King','Michael Crichton'] ? ;
Country = 'Zambia',
Authors = ['Wilbur Smith'] ? ;
no
```

```
country_authors(Country, Authors) :-
        findall(Author,
        author(_AuthorID, Author, _YearOfBirth, Country), Authors).
```

```
country_authors(Country, Authors):-
        bagof(Author, (AuthorID, YearOfBirth)^author(AuthorID, Author, YearOfBirth, Country), Authors).

% Note that we need to use bagof (or setof) to separate the results by country when the country is not instantiated.
% When using bagof (or setof), we need to use the existencial quantifier to ignore variations in AuthorID and year of birth.
% (see slides 7..9 from slides PL5)
```

Pergunta 8

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

The Passionate Fans of Literature (PFL) Book Club keeps a record of all books read by its members, as exemplified in the following code:

PT

```
read_book(bernardete, 'The Firm').
read_book(bernardete, 'The Client').
read_book(clarice, 'The Firm').
read_book(clarice, 'Carrie').
read_book(deirdre, 'The Firm').
read_book(deirdre, 'Next').
```

A book can be considered popular if it has been read by at least 75% of all the members of the PFL Book Club.
Implement ***popular(?Title)***, which unifies ***Title*** with the title of a book considered popular. Example:

```
| ?- popular('The Client').
no
| ?- popular(Title).
Title = 'The Firm' ? ;
no
```

```
popular(Title) :-
    book(Title, _AuthorID, _YearOfRelease, _Pages, _Genres),
    findall(Person, (read_book(Person, Title)), Populer),
    sort(Populer, Popular), length(Popular, P),
    findall(Peeps, (read_book(Peeps, _)), Every),
    sort(Every, Everyone), length(Everyone, E),
    Comp is 0.75*E,
    P >= Comp.
```

```
popular(Title):-
    setof(Person, _B^read_book(Person, _B), People),
    length(People, NTotal),
    setof(Person, read_book(Person, Title), HaveRead),
    length(HaveRead, N),
    N / NTotal >= 0.75.

% Note the use of setof to separate results by Title when Title is not instantiated.
```

Pergunta 9

Consider the following code:

Incorreta

Pontuou -0,150 de 0,600

Destacar pergunta

PT

Consider the following code:

```
predX(_, [], [], []).
predX(R, [A|B], [C|D], [E|F]):-
    S =.. [R,A,C,E], S,
    predX(R, B, D, F).
```

To which predicate of the lists library is *predX/4* equivalent?

a. `select/4`

b. `scanlist/4`

c. `cumlist/4` ✖

d. `maplist/4`

e. `map_product/4`

Resposta correta:
`maplist/4`

Pergunta 10

Incorreta

Pontuou -0,150 de 0,600

Retirar destaque

PT

Consider the existence of a *similarity(+Title1, +Title2, ?Similarity)* predicate that returns a measure of similarity between two books, where *Similarity* is a value between 0 and 1, and the higher the value, the more similar the books are. Consider also the following implementation of a ***most_similar(+Book, +OtherBook1, +OtherBook2, ?MostSimilar)*** predicate, which receives three book titles and unifies the fourth argument with either the second or third argument, whichever is more similar to the first book.

```
most_similar(Book, C1, C2, C1):-
    similarity(Book, C1, S1),
    similarity(Book, C2, S2),
    S1 >= S2, !.
most_similar(Book, C1, C2, C2):-
    similarity(Book, C1, S1),
    similarity(Book, C2, S2),
    S2 >= S1.
```

Which option is true regarding the cut in the code?

a. We require the implementation of *similarity/3* to determine whther the cut is red or green.

b. The cut can be considered red or green depending on the results of the call to *similarity/3*.

c. The cut is neither red nor green; it is unnecessary and can be safely removed.

d. The cut is red, as its removal would affect the results produced by most_similar/4.

e. The cut is green, as its removal would not affect results but would affect performance. ✖

Resposta correta:
The cut is red, as its removal would affect the results produced by most_similar/4.

Pergunta 11

Incorreta

Pontuou -0,150 de 0,600

Retirar destaque

PT

Consider the existence of a *predY/2* predicate with several clauses and the following code:

```
predZ(R, A, C):-
    T =.. [R, _, _],
    retract(( T:- S )),
    E =.. [R, A, C], E,
    asserta(( T:- S )).
```

What does a call to *predZ(predY, A, B)* do?

a. It replaces the arguments of the first clause of *predY/2* with the ones received by *predZ/3* as the second and third arguments. ✖

b. It changes the order of the clauses of *predY/2*.

c. It changes the order of the goals in each clause of *predY/2*.

d. It simulates a call to *predY/2* without considering its first clause.

e. Nothing, any call to *predZ/3* will always fail.

Resposta correta:
It simulates a call to *predY/2* without considering its first clause.

Pergunta 12

Respondida

Pontuou 0,400 de 1,200

Retirar destaque

PT

We want to be able to write queries and code in the format '*Author wrote Book at Age*'. Example:

```
| ?- Author wrote 'The Firm' at Age.
Author = 'John Grisham',
Age = 36 ?
```

Declare the necessary operators and code to make it possible. In a parse tree, the *wrote* operator should be above the *at* operator.

```
:- op(_, xfy, wrote).
Author wrote Book at Age :-
    author(AuthorID, Author, YearOfBirth, _CountryOfBirth),
    book(Book, AuthorID, YearOfRelease, _Pages, _Genres).
    Age is YearOfRelease - YearOfBirth.
```

```
:-op(730, xfx, wrote).
:-op(720, xfx, at).

Author wrote Book at Age:-
        author_wrote_book_at_age(Author, Book, Age).

% We first need to define the operators; note that wrote has a higher value of precedence to stay above the at operator in a parse tree.
% Then we need to define a rule that uses the new operators; sentences in this format have the same meaning as the author_wrote_book_at_age/3 predicate.
% This is similar to the example in slide 20 of slides PL7 and exercises 7 and 8 of exercise sheet P06.
```

Pergunta 13

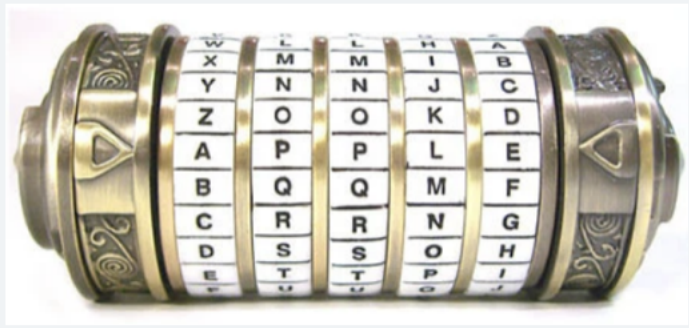
Respondida

Pontuou 1,500 de 1,500

Destacar pergunta

PT

Consider a codex, like the one shown in the figure, represented by a list of lists, as shown below. Consider that all the lists have the same length.



```
[ [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],
  [p, q, r, s, t, u, v, w, x, y, z, 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, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o],
  [l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, a, b, c, d, e, f, g, h, i, j, k],
  [e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, a, b, c, d] ]
```

Implement ***rotate(+List, +Position, +Rotations, ?NewList)***, which receives a ***List*** with the initial configuration of the codex, the ***Position*** of the dial to rotate (consider that indexes start at 1), and the number of ***Rotations*** to be performed, unifying ***NewList*** with the resulting codex configuration. The predicate should fail if the received Position is outside the valid range. Both negative numbers and numbers larger than the size of the dial can be accepted. Examples:

```
| ?- rotate([ [3,6,8], [5,0,7], [2,1,4] ], 2, 1, NewList).
NewList = [ [3,6,8], [0,7,5], [2,1,4] ]
| ?- rotate([ [3,6,8], [5,0,7], [2,1,4] ], 3, 2, NewList).
NewList = [ [3,6,8], [5,0,7], [4,2,1] ]
| ?- rotate([ [3,6,8], [5,0,7], [2,1,4] ], 1, -1, NewList).
NewList = [ [8,3,6], [5,0,7], [2,1,4] ]
| ?- rotate([ [3,6,8], [5,0,7], [2,1,4] ], 1, 4, NewList).
NewList = [ [6,8,3], [5,0,7], [2,1,4] ]
```

```
:- use_module(library(lists)).
rotate(List, Position, Rotations, NewList) :-
    nth1(Position, List, Res),
    append(Prefix, [Res|Suffix], List),
    rotate_list(Rotations, Res, Rotated),
    append(Prefix, [Rotated|Suffix], NewList).
```

```
rotate(Code, Index, Amount, NewCode):-
        nth1(Index, Code, List, Rest),
        rotate_list(Amount, List, Rotated),
        nth1(Index, NewCode, Rotated, Rest).

% The nth1/4 predicate is used to access the dial to rotate, and then replace it with the changed one
% The rotate_list/3 predicate is used to rotate the dial
% (see slides 23 and 26 from slides PL3 and the lists library documentation)
```

```
% The rotate_list/3 predicate is used to rotate the dial
% (see slides 23 and 26 from slides PL3 and the lists library documentation)
```

Pergunta 14

Respondida

Pontuou 1,400 de 1,400

Destacar pergunta

Implement ***matches(+List, +Code)***, which succeeds if the codex ***List*** is aligned with ***Code*** as a solution (i.e., each digit of ***Code*** corresponds to the first digit of the list in the corresponding position of the codex), and fails otherwise. The predicate must fail if ***Code*** is different in size from the number of elements in the codex ***List***. Example:

PT

```
| ?- matches([ [3,6,8], [5,0,7], [2,1,4] ], [3,5,2]).
yes
| ?- matches([ [3,6,8], [5,0,7], [2,1,4] ], [6,0,1]).
no
| ?- matches([ [3,6,8], [5,0,7], [2,1,4] ], [3,5,2,1]).
no
```

```
:- use_module(library(lists)).
matches([H|T], [HC|RC]) :-
    nth1(Value, H, HC),
    matches_rec(T, RC, Value), !.
matches_rec([], [], _).
```

```
matches([], []).
matches([[H|_] | Code], [H|Key]):-
        matches(Code, Key).

% Simple list iteration to check the code
```

Pergunta 15

Respondida

Pontuou 1,300 de 1,400

Destacar pergunta

Implement ***move(+Initial, -Final)***, which returns in ***Final***, one by one via backtracking, all the possible moves for the given ***Initial*** codex configuration (possible rotations for each of the dials). Example:

PT

```
| ?- move([[3,6,8],[5,0,7],[2,1,4]], Final).
Final = [[6,8,3],[5,0,7],[2,1,4]] ? n
Final = [[8,3,6],[5,0,7],[2,1,4]] ? n
Final = [[3,6,8],[0,7,5],[2,1,4]] ? n
Final = [[3,6,8],[7,5,0],[2,1,4]] ? n
Final = [[3,6,8],[5,0,7],[1,4,2]] ? n
Final = [[3,6,8],[5,0,7],[4,2,1]] ? n
no
```

```
:- use_module(library(lists)).
rotate(List, Position, Rotations, NewList) :-
    nth1(Position, List, Res),
    append(Prefix, [Res|Suffix], List),
    rotate_list(Rotations, Res, Rotated),
    append(Prefix, [Rotated|Suffix], NewList).

:-use_module(library(between)).
move(Initial, Final) :-
    length(Initial, L),
```

```
move(Code, NewCode):-
        length(Code, Len),
        nth1(1, Code, List),
        length(List, Size),
        Max is Size - 1,
        between(1, Len, Index),
        between(1, Max, Amount),
        rotate(Code, Index, Amount, NewCode).

% Note that the size of the codex is not necessarily the same as the size of the dials.
% between/3 is used to generate all possible move configurations via backtracking.
```

Pergunta 16

Respondida

Pontuou 0,400 de 1,500

Destacar pergunta

Implement ***solve(+Code, +Key, -States)***, which returns in ***States*** the set of states through which the codex passes to be solved, avoiding repeated states (States must contain the initial ***Code***, the intermediate ones, and the final one). ***Key*** contains the code (representing the first element of each list of the solution). Solutions that return shorter paths will be valued.

PT

Examples:

```
| ?- solve([[3,6,8],[5,0,7],[2,1,4]], [6,0,4], States).
States = [[[3,6,8],[5,0,7],[2,1,4]], [[6,8,3],[5,0,7],[2,1,4]], [[6,8,3],[0,7,5],[2,1,4]], [[6,8,3],[0,7,5],[4,2,1]]] ?
yes
| ?- solve([[3,6,8],[5,0,7],[2,1,4]], [6,5,1], States).
States = [[[3,6,8],[5,0,7],[2,1,4]], [[6,8,3],[5,0,7],[2,1,4]], [[6,8,3],[5,0,7],[1,4,2]]] ?
yes
```

```
:-use_module(library(random)).
:- use_module(library(lists)).
:-use_module(library(between)).

move_random(Initial, Final) :-
    length(Initial, L),
    Rot is L - 1, Rot > 0,
    random(1, L, Position),
    random(1, Rot, Rotations),
    rotate(Initial, Position, Rotations, Final).
```

```
solve(Code, Key, States):-
        solveb([[Code]], Key, States).

solveb([[Code|Rest]|_], Key, States):-
        matches(Code, Key), !,
        reverse([Code|Rest], States).
solveb([[Code|Path]|Rest], Key, States):-
        findall( Next, ( move(Code, Next), \+ member(Next, [Code|Path]) ), NextCodes),
        add_all(NextCodes, [Code|Path], New),
        append(Rest, New, NextQ),
        solveb(NextQ, Key, States).

add_all([], L, L).
add_all([Path|Rest], Old, [[Path|Old]|T]):-
        add_all(Rest, Old, T).

% This solution uses a breadth-first search (see slide 18 of slides PL5) to reach the solution
```