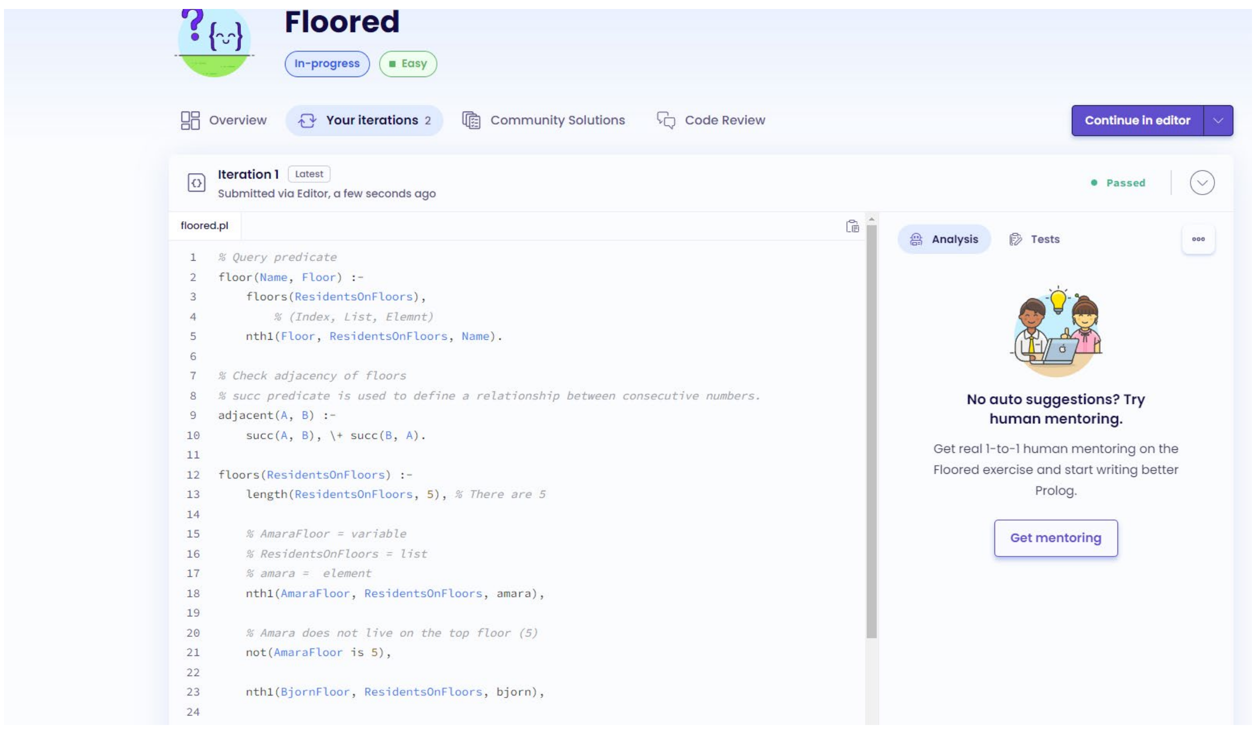
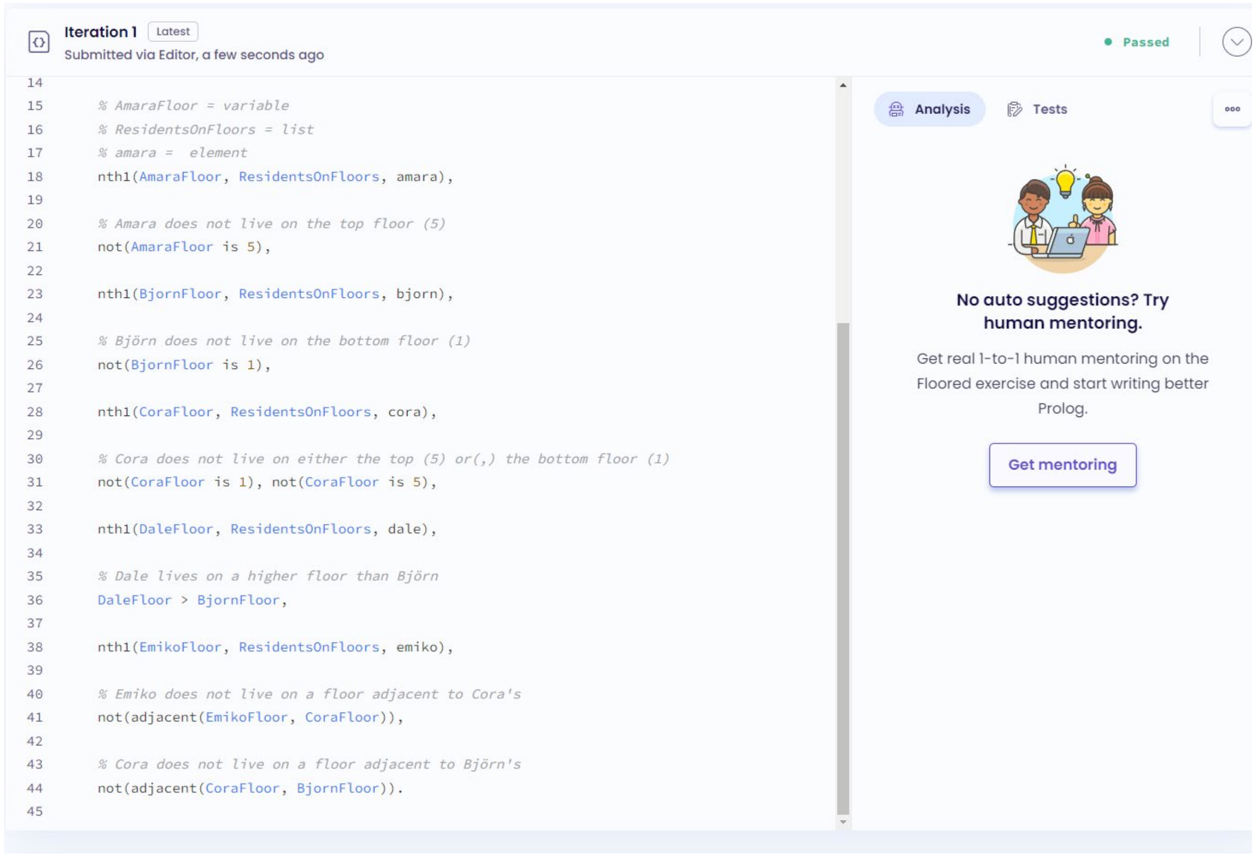
1. **Floored:**

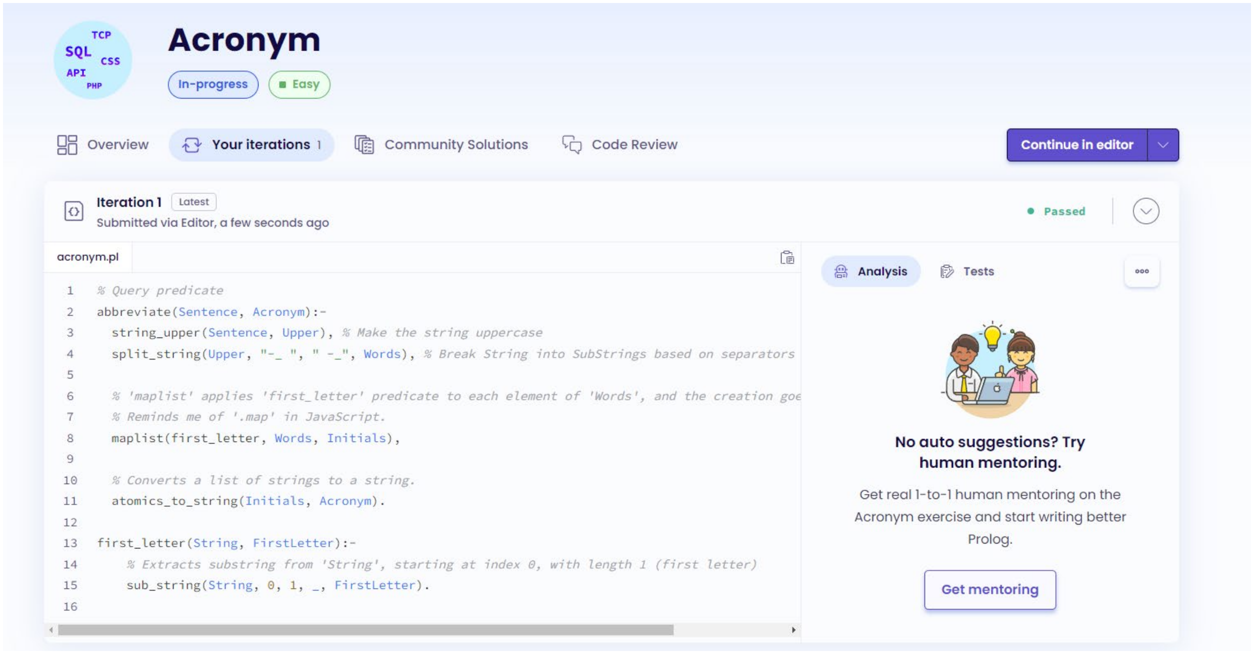

Floored
In-progress
Easy
Overview
Your iterations 2
Community Solutions
Code Review
Continue in editor
Iteration 1
Latest
Submitted via Editor, a few seconds ago
Passed
floored.pl
Analysis
Tests
No auto suggestions? Try human mentoring.
Get real 1-to-1 human mentoring on the Floored exercise and start writing better Prolog.
Get mentoring

```prolog
% Query predicate
floor(Name, Floor) :-
    floors(ResidentsOnFloors),
        % (Index, List, Elemnt)
    nth1(Floor, ResidentsOnFloors, Name).

% Check adjacency of floors
% succ predicate is used to define a relationship between consecutive numbers.
adjacent(A, B) :-
    succ(A, B), \+ succ(B, A).

floors(ResidentsOnFloors) :-
    length(ResidentsOnFloors, 5), % There are 5

    % AmaraFloor = variable
    % ResidentsOnFloors = list
    % amara =  element
    nth1(AmaraFloor, ResidentsOnFloors, amara),

    % Amara does not live on the top floor (5)
    not(AmaraFloor is 5),

    nth1(BjornFloor, ResidentsOnFloors, bjorn),

```


Iteration 1
Latest
Submitted via Editor, a few seconds ago
Passed
Analysis
Tests
No auto suggestions? Try human mentoring.
Get real 1-to-1 human mentoring on the Floored exercise and start writing better Prolog.
Get mentoring

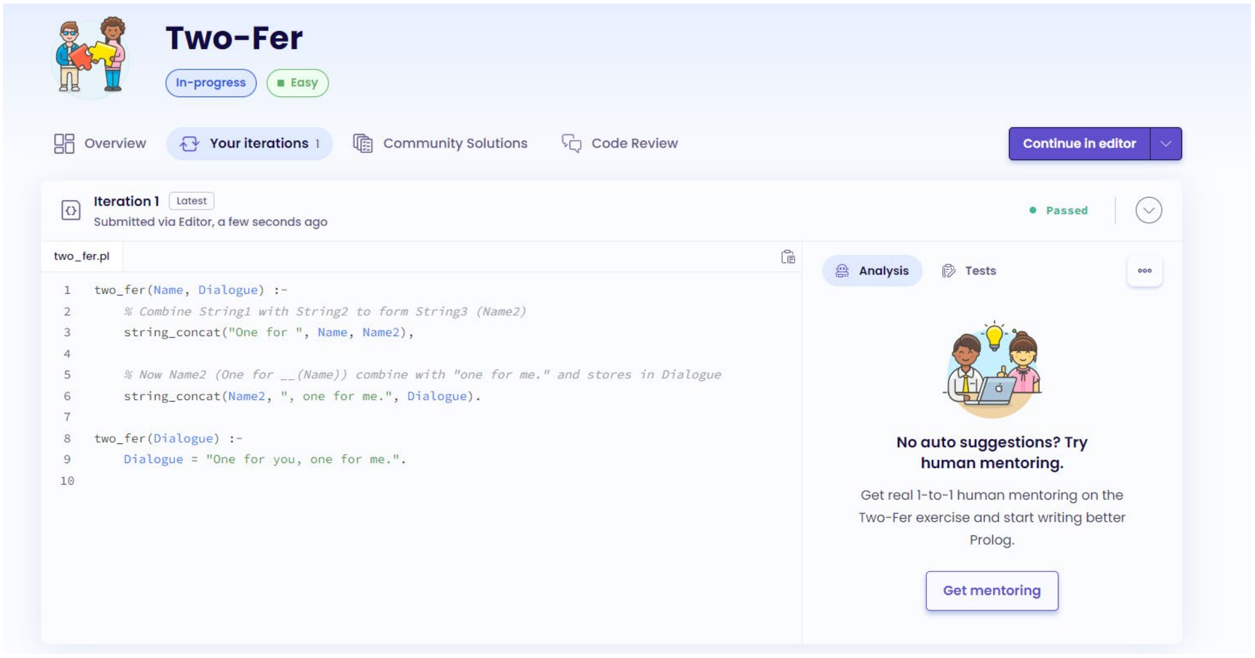
```prolog

    % AmaraFloor = variable
    % ResidentsOnFloors = list
    % amara =  element
    nth1(AmaraFloor, ResidentsOnFloors, amara),

    % Amara does not live on the top floor (5)
    not(AmaraFloor is 5),

    nth1(BjornFloor, ResidentsOnFloors, bjorn),

    % Björn does not live on the bottom floor (1)
    not(BjornFloor is 1),

    nth1(CoraFloor, ResidentsOnFloors, cora),

    % Cora does not live on either the top (5) or(,) the bottom floor (1)
    not(CoraFloor is 1), not(CoraFloor is 5),

    nth1(DaleFloor, ResidentsOnFloors, dale),

    % Dale lives on a higher floor than Björn
    DaleFloor > BjornFloor,

    nth1(EmikoFloor, ResidentsOnFloors, emiko),

    % Emiko does not live on a floor adjacent to Cora's
    not(adjacent(EmikoFloor, CoraFloor)),

    % Cora does not live on a floor adjacent to Björn's
    not(adjacent(CoraFloor, BjornFloor)).

```

2. **Acronym**


Acronym
In-progress
Easy
Overview
Your iterations 1
Community Solutions
Code Review
Continue in editor
Iteration 1
Latest
Submitted via Editor, a few seconds ago
Passed
acronym.pl
Analysis
Tests
No auto suggestions? Try human mentoring.
Get real 1-to-1 human mentoring on the Acronym exercise and start writing better Prolog.
Get mentoring

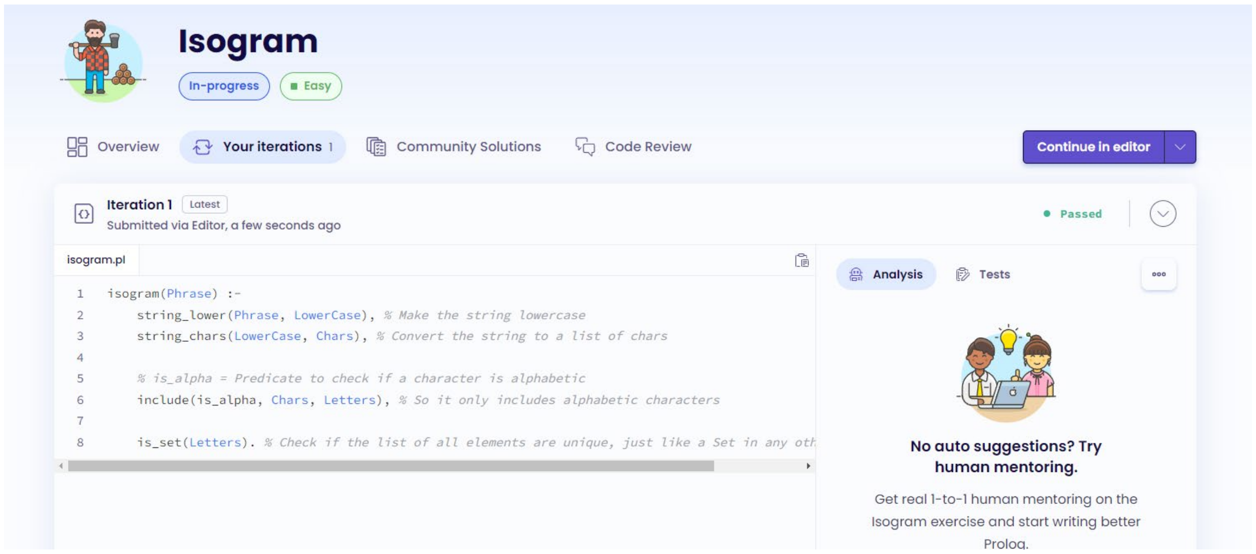
```prolog
% Query predicate
abbreviate(Sentence, Acronym):-
  string_upper(Sentence, Upper), % Make the string uppercase
  split_string(Upper, "-_ ", " -_", Words), % Break String into SubStrings based on separators

  % 'maplist' applies 'first_letter' predicate to each element of 'Words', and the creation goe
  % Reminds me of '.map' in JavaScript.
  maplist(first_letter, Words, Initials),

  % Converts a list of strings to a string.
  atomics_to_string(Initials, Acronym).

first_letter(String, FirstLetter):-
    % Extracts substring from 'String', starting at index 0, with length 1 (first letter)
    sub_string(String, 0, 1, _, FirstLetter).

```

3. **Two-Fer**


Two-Fer
In-progress
Easy
Overview
Your iterations 1
Community Solutions
Code Review
Continue in editor
Iteration 1
Latest
Submitted via Editor, a few seconds ago
Passed
two_fer.pl
Analysis
Tests
No auto suggestions? Try human mentoring.
Get real 1-to-1 human mentoring on the Two-Fer exercise and start writing better Prolog.
Get mentoring

```prolog
two_fer(Name, Dialogue) :-
    % Combine String1 with String2 to form String3 (Name2)
    string_concat("One for ", Name, Name2),

    % Now Name2 (One for __(Name)) combine with "one for me." and stores in Dialogue
    string_concat(Name2, ", one for me.", Dialogue).

two_fer(Dialogue) :-
    Dialogue = "One for you, one for me.".

```

4. **Isogram**



```
isogram(Phrase) :-
    string_lower(Phrase, LowerCase), % Make the string lowercase
    string_chars(LowerCase, Chars), % Convert the string to a list of chars

    % is_alpha = Predicate to check if a character is alphabetic
    include(is_alpha, Chars, Letters), % So it only includes alphabetic characters

    is_set(Letters). % Check if the list of all elements are unique, just like a Set in any oth
```

No auto suggestions? Try human mentoring.

Get real 1-to-1 human mentoring on the Isogram exercise and start writing better Prolog.