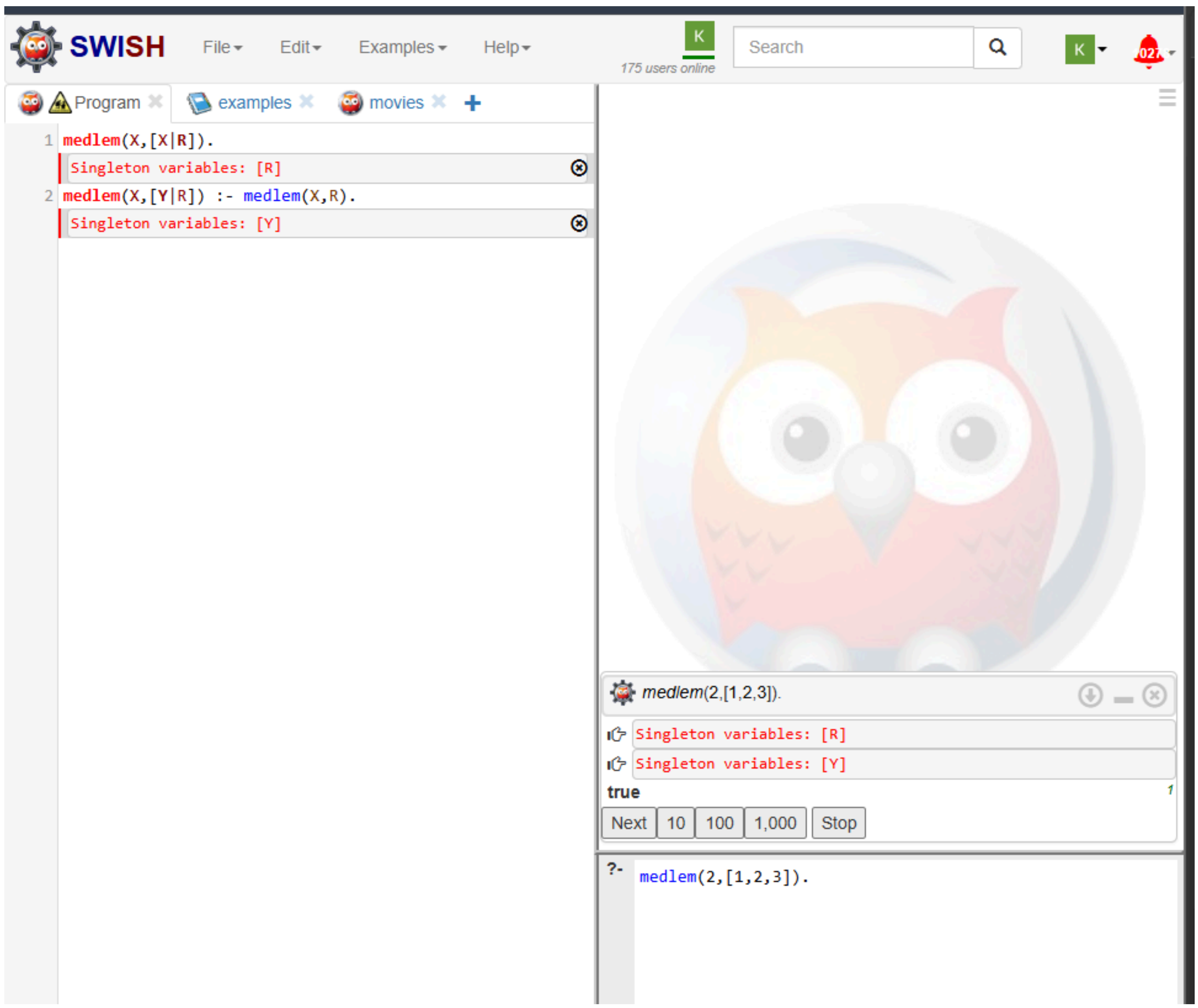
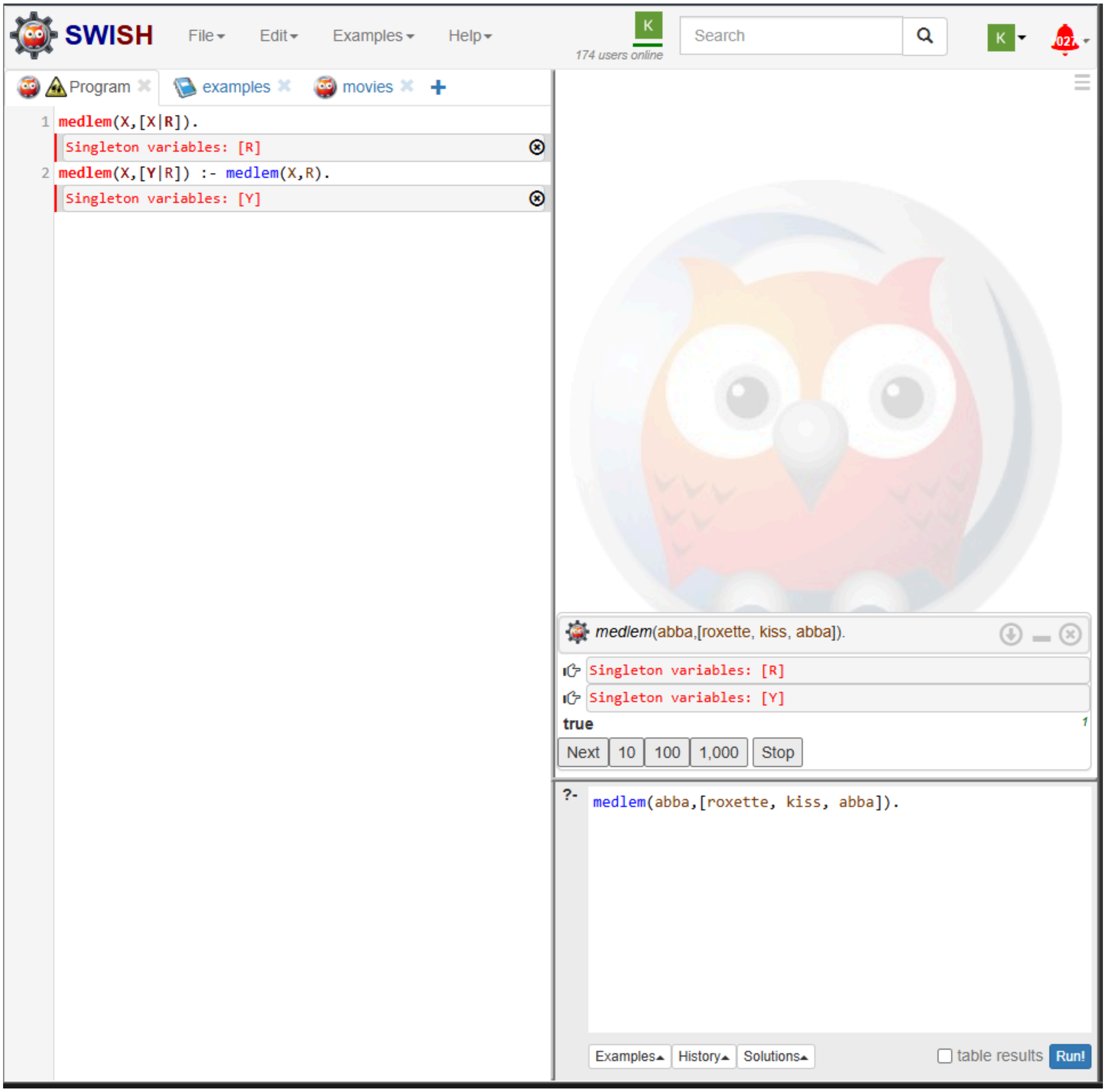
DVA-265 Artificial Intelligence 2
LAB - 01

Khalid Hasan Ador and Pauline Laval

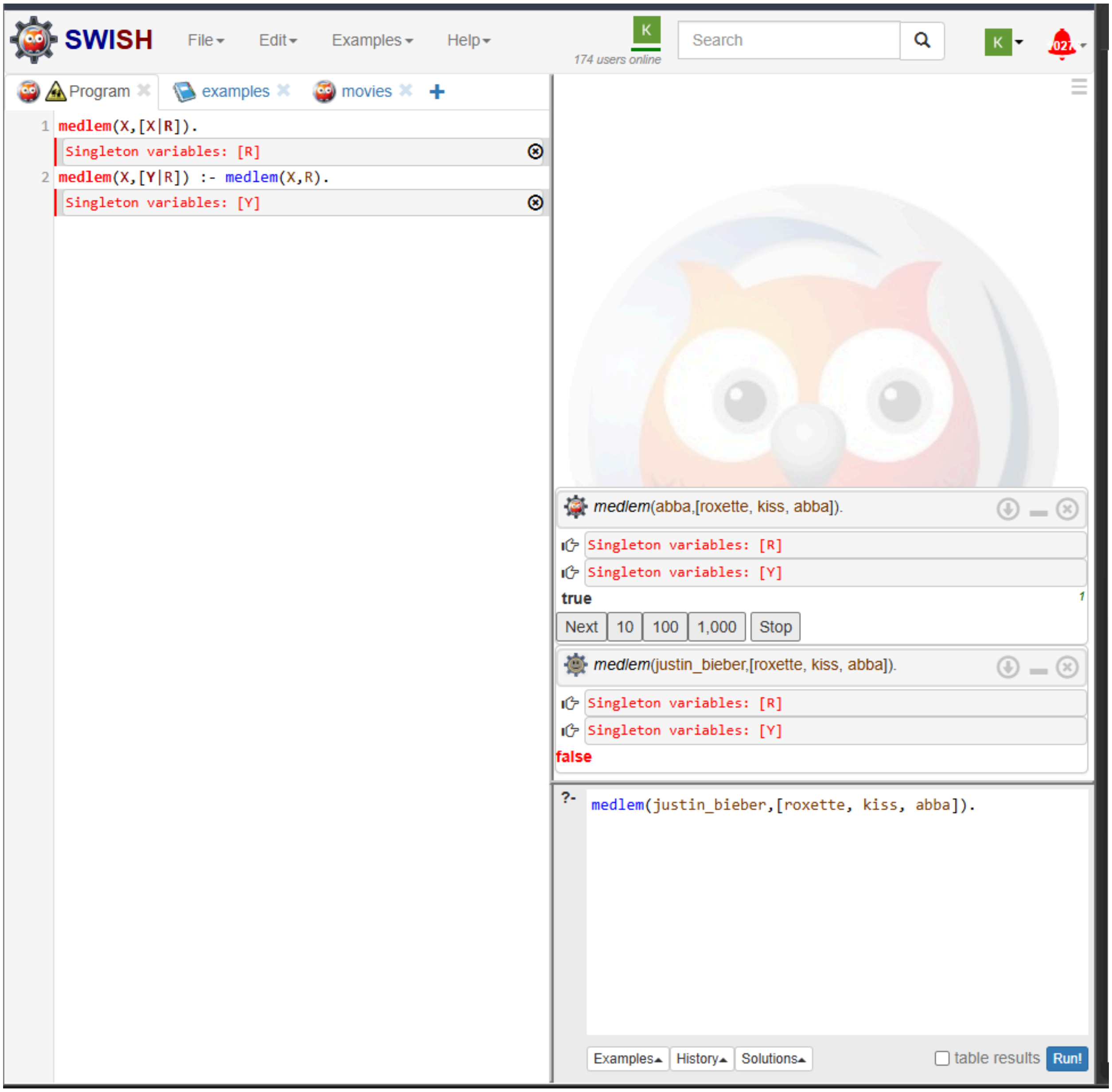
1) medlem(2,[1,2,3])
   True



2) medlem(abba,[roxette, kiss, abba]).
True

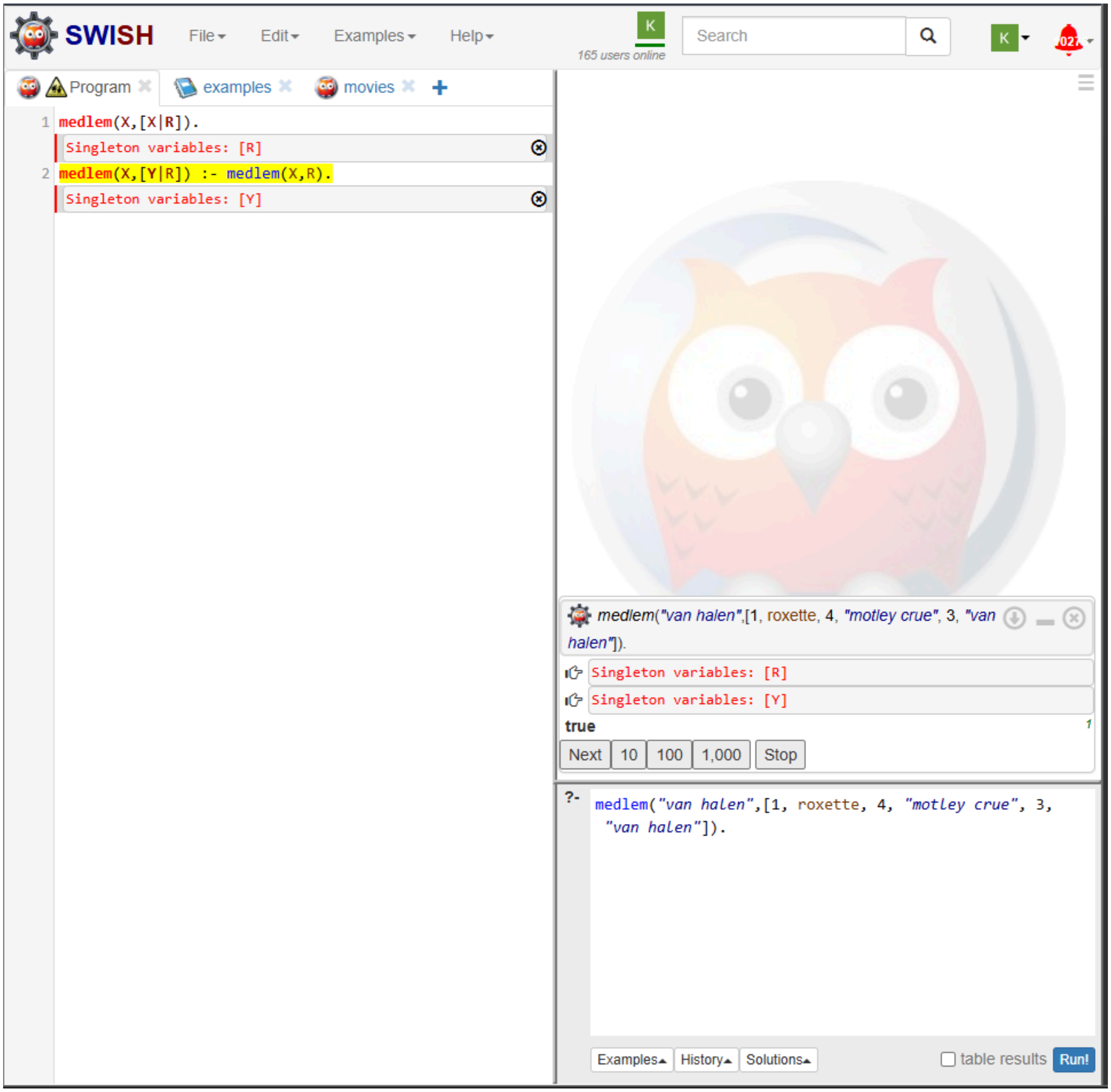


medlem(justin_bieber,[roxette, kiss, abba]).
Flase



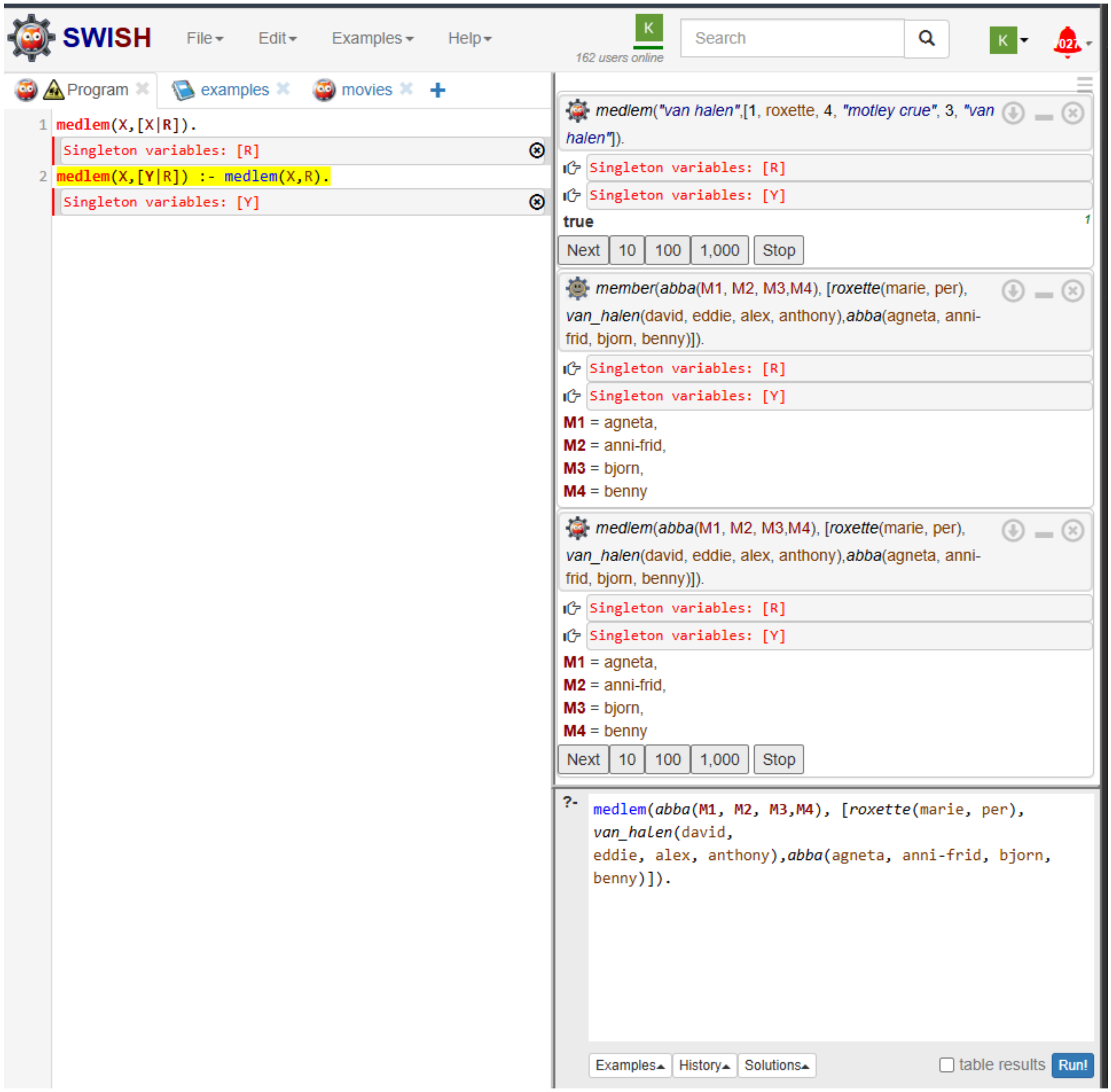
Palin Language: "Is abba/justin_bieber one of the items in the list?"

3) medlem(”van halen”,[1, roxette, 4, ”motley crue”, 3, ”van halen”]).
True



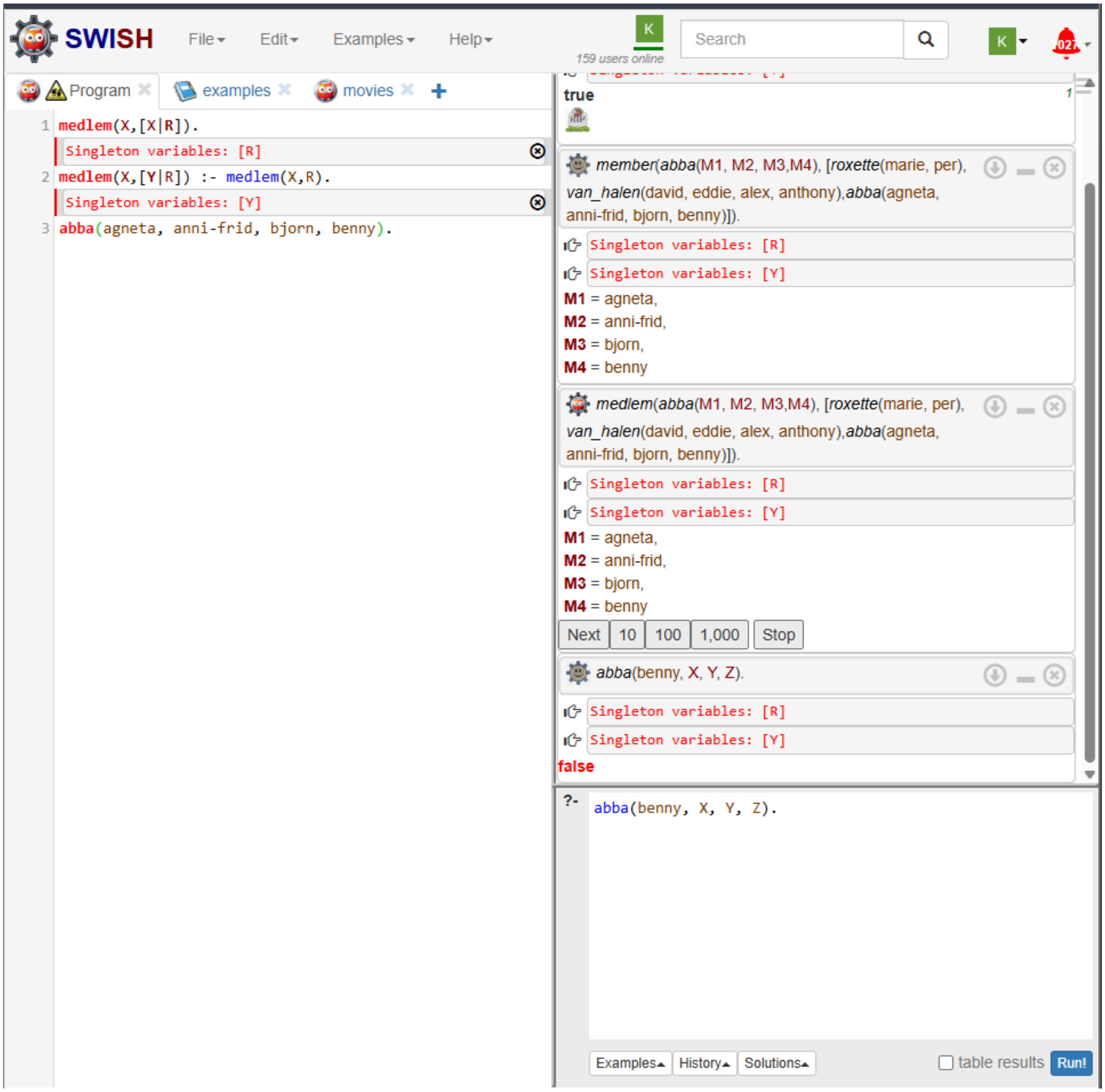
N.B.: Don’t use curly quotes

4) member(abba(M1, M2, M3,M4), [roxette(marie, per), van_halen(david, eddie, alex, anthony),abba(agneta, anni-frid, bjorn, benny)]).



N>B: This demonstrates Prolog's unification: it matches the structure abba(...) in the list and binds the variables.

5) abba(benny, X, Y, Z).



Prolog tries to unify abba(benny, X, Y, Z) with abba(agneta, anni-frid, bjorn, benny).
Since benny ≠ agneta, unification fails.

6) **largest_element(X, [X]).**
This says that if the list has only one element (i.e., `[X]`), element `X` is the largest by default. This stops the recursion.

largest_element(X, [X|Rest]):- largest_element(Y, Rest), X >=Y.

Recursive case 1 (X is greater than or equal to the largest in the rest):

- The list starts with the largest
- The list is of the form `[X|Rest]`.
- First, it finds the largest element `Y` in the `Rest` of the list.
- Then it checks if `X >= Y`.
- If so, it concludes that `X` is the largest element in the entire list.

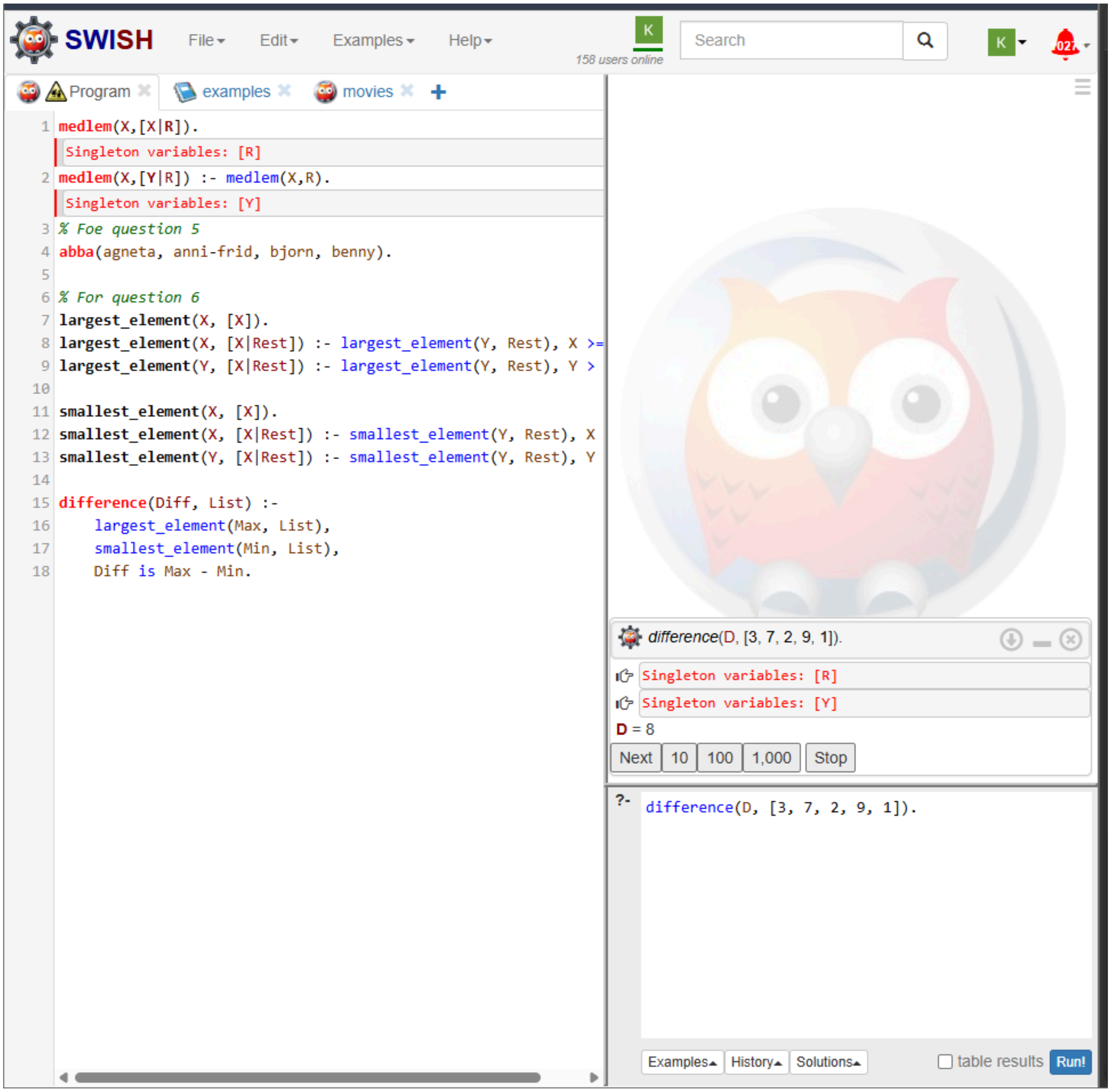
largest_element(N, [X|Rest]):- largest_element(N, Rest), N > X.

Recursive case 2 (largest in Rest is greater than X):

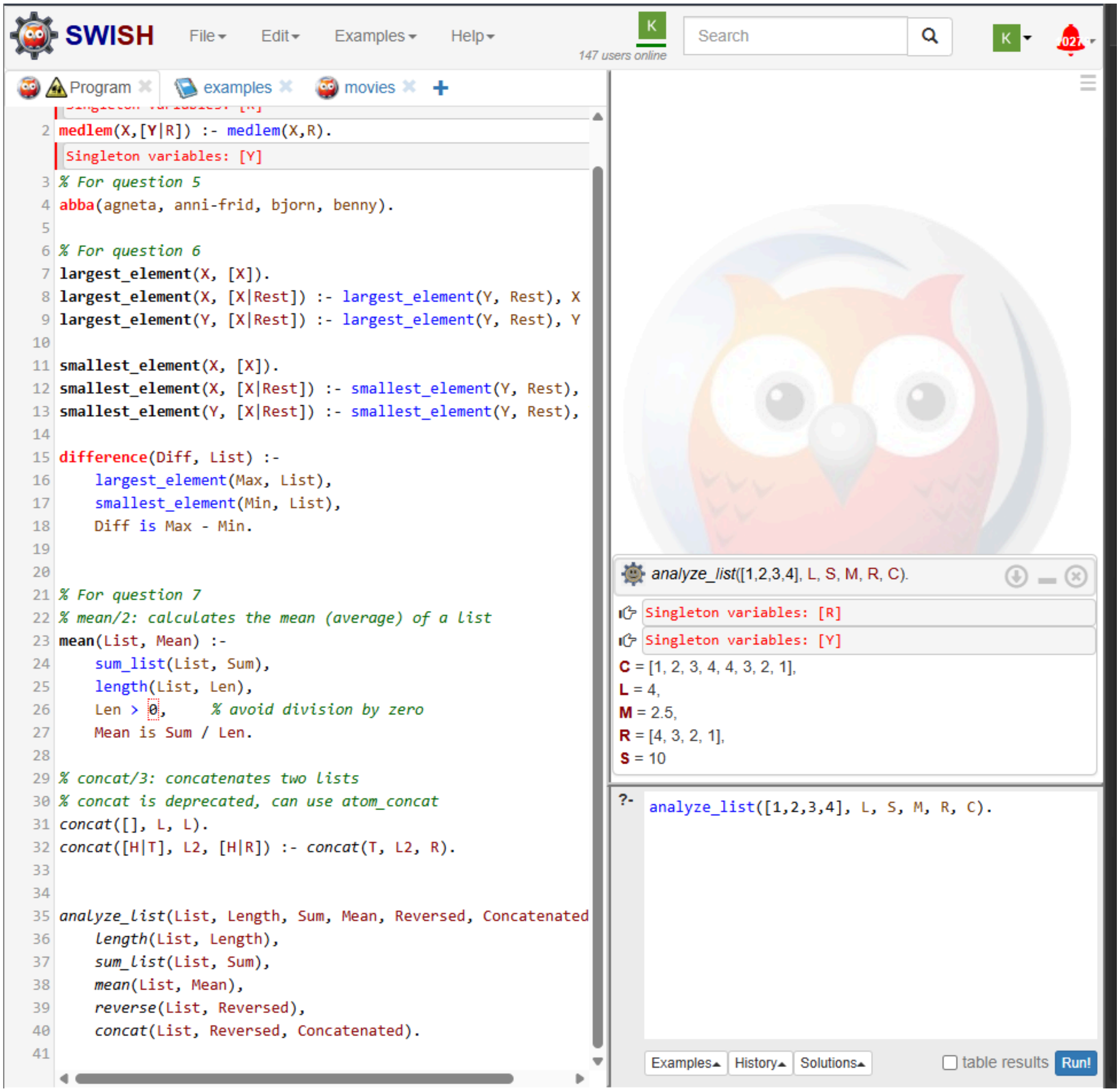
- The largest is in the rest
- Again, the list is `[X|Rest]`.
- It finds the largest element `N` in the `Rest`.
- If `N > X`, then `N` is the largest element overall.

```
smallest_element(X, [X]).
smallest_element(X, [X|Rest]) :- smallest_element(Y, Rest), X =< Y.
smallest_element(Y, [X|Rest]) :- smallest_element(Y, Rest), Y < X.
```

```
difference(Diff, List) :-
        largest_element(Max, List),
        smallest_element(Min, List),
        Diff is Max - Min.
```


SWISH
File
Edit
Examples
Help
158 users online
Search
Program
examples
movies
medlem(X,[X|R]).
Singleton variables: [R]
medlem(X,[Y|R]) :- medlem(X,R).
Singleton variables: [Y]
% Foe question 5
abba(agneta, anni-frid, bjorn, benny).
% For question 6
largest_element(X, [X]).
largest_element(X, [X|Rest]) :- largest_element(Y, Rest), X >=
largest_element(Y, [X|Rest]) :- largest_element(Y, Rest), Y >
smallest_element(X, [X]).
smallest_element(X, [X|Rest]) :- smallest_element(Y, Rest), X
smallest_element(Y, [X|Rest]) :- smallest_element(Y, Rest), Y
difference(Diff, List) :-
largest_element(Max, List),
smallest_element(Min, List),
Diff is Max - Min.
difference(D, [3, 7, 2, 9, 1]).
Singleton variables: [R]
Singleton variables: [Y]
D = 8
Next
10
100
1,000
Stop
?-
difference(D, [3, 7, 2, 9, 1]).
Examples
History
Solutions
table results
Run!

7)


SWISH
File
Edit
Examples
Help
147 users online
Search
Program
examples
movies
medlem(X,[Y|R]) :- medlem(X,R).
Singleton variables: [Y]
% For question 5
abba(agneta, anni-frid, bjorn, benny).
% For question 6
largest_element(X, [X]).
largest_element(X, [X|Rest]) :- largest_element(Y, Rest), X
largest_element(Y, [X|Rest]) :- largest_element(Y, Rest), Y
smallest_element(X, [X]).
smallest_element(X, [X|Rest]) :- smallest_element(Y, Rest),
smallest_element(Y, [X|Rest]) :- smallest_element(Y, Rest),
difference(Diff, List) :-
largest_element(Max, List),
smallest_element(Min, List),
Diff is Max - Min.
% For question 7
% mean/2: calculates the mean (average) of a list
mean(List, Mean) :-
sum_list(List, Sum),
length(List, Len),
Len > 0, % avoid division by zero
Mean is Sum / Len.
% concat/3: concatenates two lists
% concat is deprecated, can use atom_concat
concat([], L, L).
concat([H|T], L2, [H|R]) :- concat(T, L2, R).
analyze_list(List, Length, Sum, Mean, Reversed, Concatenated
length(List, Length),
sum_list(List, Sum),
mean(List, Mean),
reverse(List, Reversed),
concat(List, Reversed, Concatenated).
analyze_list([1,2,3,4], L, S, M, R, C).
Singleton variables: [R]
Singleton variables: [Y]
C = [1, 2, 3, 4, 4, 3, 2, 1],
L = 4,
M = 2.5,
R = [4, 3, 2, 1],
S = 10
?- analyze_list([1,2,3,4], L, S, M, R, C).
Examples
History
Solutions
table results
Run!

8)

8.1 Addition:

Base Case: peano_add(0, Y, Y).

If the first number is 0, then the result of the addition is simply Y

Recursive Case: peano_add(f(X), Y, f(Z)) :- peano_add(X, Y, Z).

If X is not 0, i.e., it is f(X1) for some X1, then:

- Add X1 and Y to get Z
- Then the final result is f(Z), which is one more than Z

8.2 Subtraction:

Base Case: peano_sub(X, 0, X).

If the second number is 0, then the result is simply the first number

Recursive Case: peano_sub(f(X), f(Y), Z) :- peano_sub(X, Y, Z).

If both numbers are successors (e.g., f(X) and f(Y)), subtract the "inner parts" X and Y.

That is: f(X) − f(Y) = X − Y

Until it matches the base case.

8.3 Multiplication:

Base Case: peano_mul(0, _, 0).

Anything multiplied by 0 results in 0.

Recursive Case:

```
peano_mul(f(X), Y, Z) :-
   peano_mul(X, Y, Z1),
   peano_add(Y, Z1, Z).
```

(X+1) * Y = Y + (X * Y)

8.4 Division:

Base Case: peano_div(X, Y, 0) :- peano_lt(X, Y). % if X < Y, quotient is zero

- If X is **less than** Y, you cannot subtract Y even once from X.
- So the quotient is 0.

Recursive Case:

peano_div(X, Y, f(Q)) :-
  peano_sub(X, Y, Z),
  peano_div(Z, Y, Q).

→ Subtract `Y` from `X` once → `Z = X - Y`.

→ Recursively divide `Z` by `Y`, getting `Q`.

→ Add one more to `Q` using the successor function `f/1` → total quotient is `f(Q)`.

8.5 Less Than:

Base Case: peano_lt(0, f(_)).

O is less than anything

Recursive Case: peano_lt(f(X), f(Y)) :- peano_lt(X, Y).

If both numbers are successors (`f(X)` and `f(Y)`), recursively compare the predecessors.

8.6 Peano to Integer:

Base Case: peano_to_int(0, 0).

0 becomes 0

Recursive Case:

peano_to_int(f(P), N) :-
  peano_to_int(P, N1),
  N is N1 + 1.

Each `f(P)` adds 1 to the result of converting `P`.