

DVA-265 Artificial Intelligence 2

LAB - 01

Khalid Hasan Ador and Pauline Laval

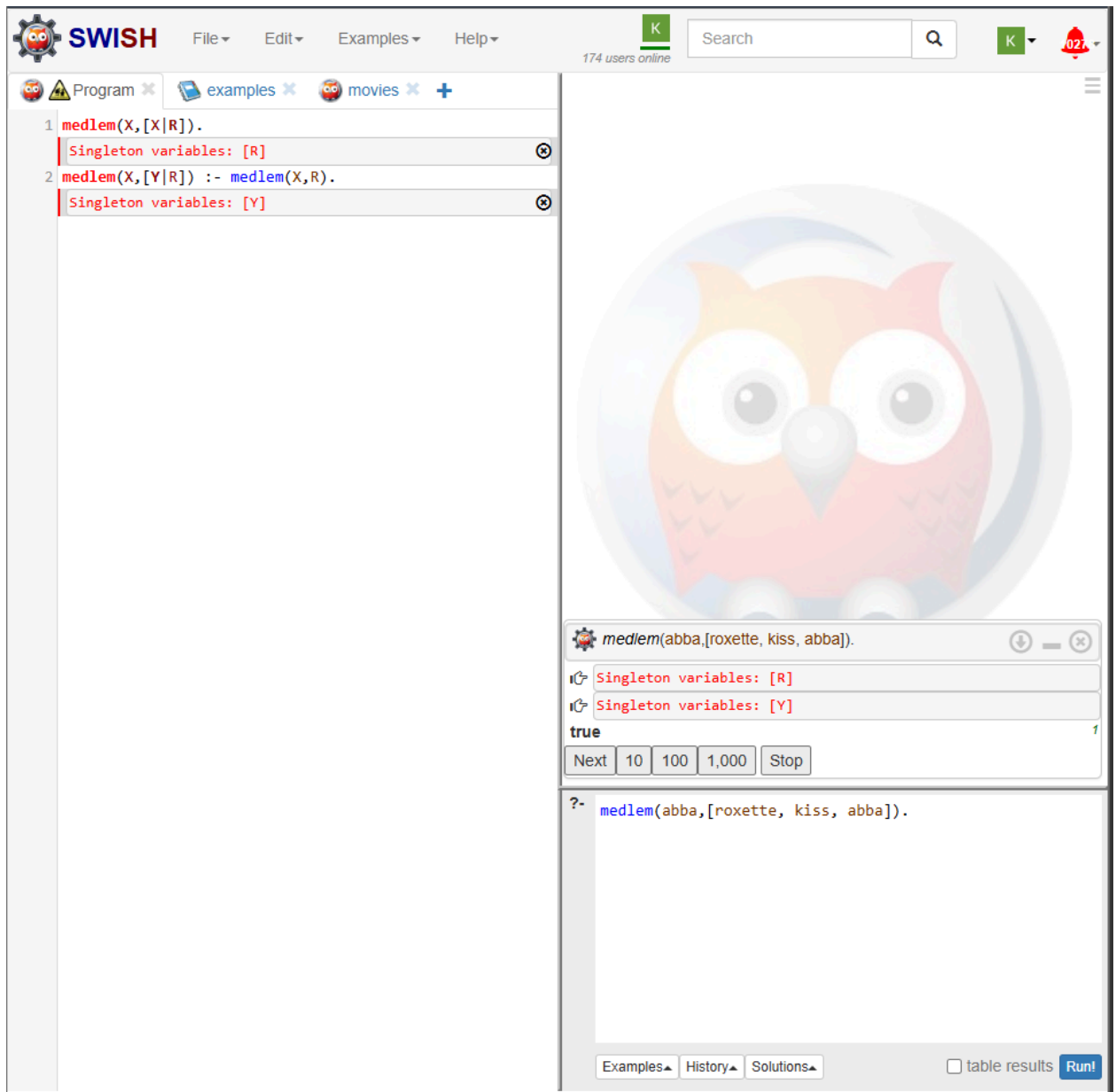
- 1) `medlem(2,[1,2,3])`
True

The screenshot displays the SWISH Prolog IDE interface. The top menu bar includes 'File', 'Edit', 'Examples', and 'Help'. A search bar is located on the right. The main editor area on the left contains the following Prolog code:

```
1 medlem(X,[X|R]).  
   Singleton variables: [R]  
2 medlem(X,[Y|R]) :- medlem(X,R).  
   Singleton variables: [Y]
```

The right pane shows a large owl logo. Below it, a console window displays the execution of the query `medlem(2,[1,2,3]).`. The output shows the singleton variables `[R]` and `[Y]`, followed by the result `true`. The console also includes navigation buttons: 'Next', '10', '100', '1,000', and 'Stop'.

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



The screenshot shows the SWISH Prolog IDE interface. The top menu bar includes 'File', 'Edit', 'Examples', and 'Help'. A search bar is on the right. The main editor area on the left contains the following Prolog code:

```
1 medlem(X,[X|R]).  
   Singleton variables: [R]  
2 medlem(X,[Y|R]) :- medlem(X,R).  
   Singleton variables: [Y]
```

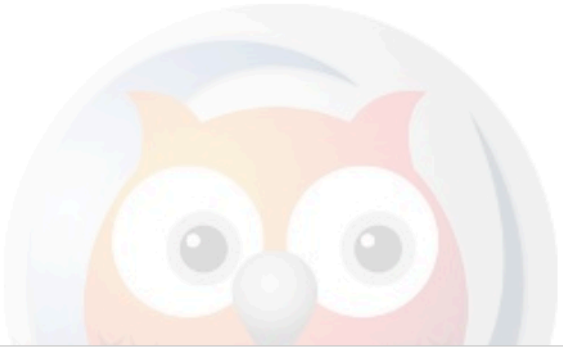
The right pane displays a large owl logo. Below it, a console window shows the execution of the query `medlem(abba,[roxette, kiss, abba]).`. The output indicates that the query is true, with singleton variables `[R]` and `[Y]`. The console also shows a 'true' result and a '1' count. At the bottom of the console, there are buttons for 'Next', '10', '100', '1,000', and 'Stop'. The bottom status bar includes 'Examples', 'History', 'Solutions', a checkbox for 'table results', and a 'Run!' button.

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

SWISH File Edit Examples Help 174 users online Search

Program examples movies +

```
1 medlem(X,[X|R]).
   Singleton variables: [R]
2 medlem(X,[Y|R]) :- medlem(X,R).
   Singleton variables: [Y]
```



medlem(abba,[roxette, kiss, abba]).
Singleton variables: [R]
Singleton variables: [Y]
true
Next 10 100 1,000 Stop

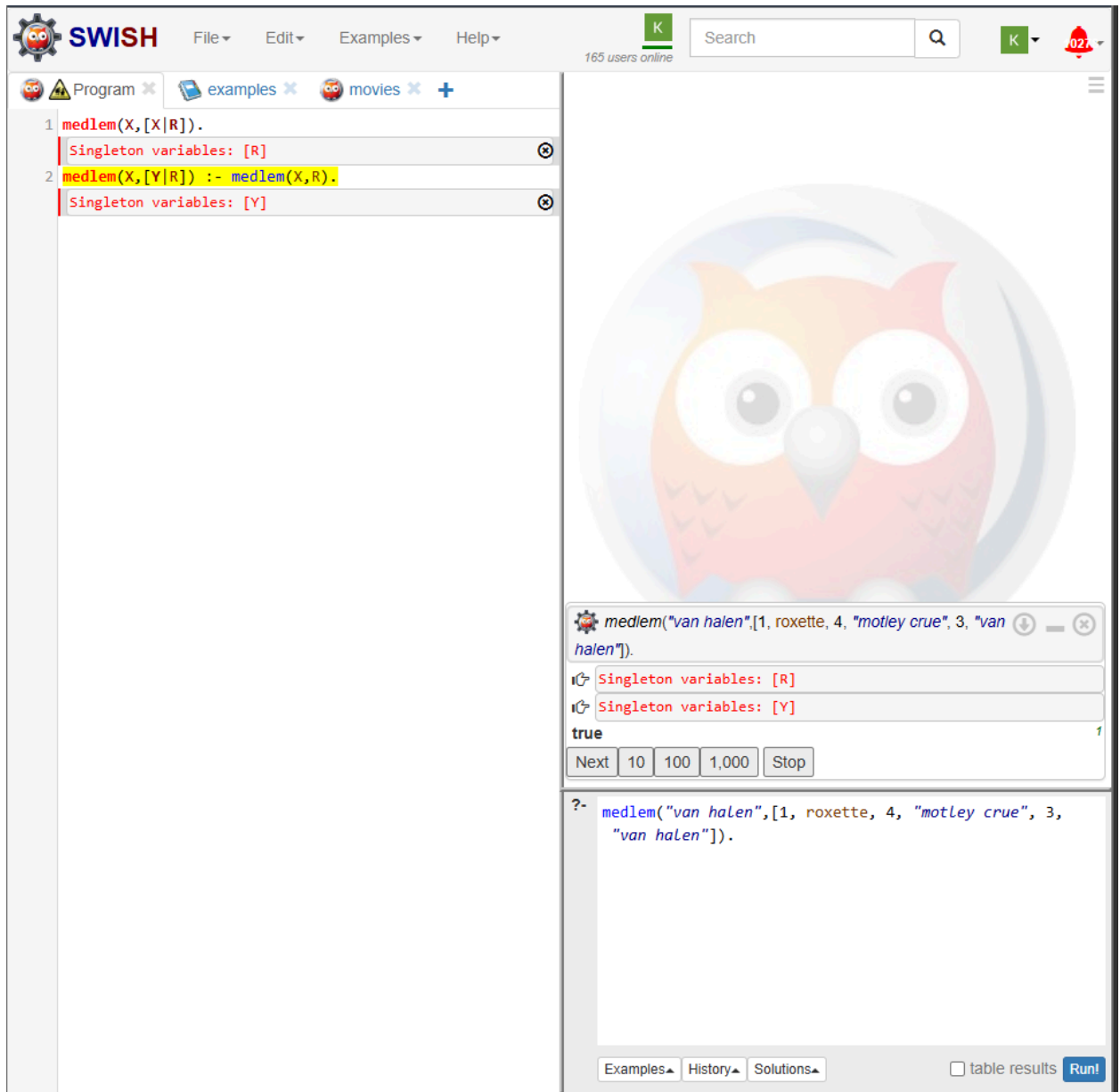
medlem(justin_bieber,[roxette, kiss, abba]).
Singleton variables: [R]
Singleton variables: [Y]
false

?- medlem(justin_bieber,[roxette, kiss, abba]).

Examples History Solutions ☐ table results Run!

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



The screenshot shows the SWISH Prolog IDE interface. The top menu bar includes "File", "Edit", "Examples", and "Help". A search bar is on the right. Below the menu, there are tabs for "Program", "examples", and "movies". The "Program" tab is active, showing a Prolog program with two lines:

```
1 medlem(X,[X|R]).  
   Singleton variables: [R]  
2 medlem(X,[Y|R]) :- medlem(X,R).  
   Singleton variables: [Y]
```

The right side of the interface features a large owl logo. Below it, a small window displays the execution of the query:

```
medlem("van halen",[1, roxette, 4, "motley crue", 3, "van halen"]).
```

Below the query, it shows the singleton variables:

```
Singleton variables: [R]  
Singleton variables: [Y]
```

The result is "true". At the bottom of this window, there are buttons for "Next", "10", "100", "1,000", and "Stop".

At the bottom of the main interface, there are buttons for "Examples", "History", and "Solutions", along with a checkbox for "table results" and a "Run!" button.

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)]).`

The screenshot shows the SWISH Prolog IDE interface. On the left, the 'Program' tab contains the following code:

```
1 medlem(X,[X|R]).
Singleton variables: [R]
2 medlem(X,[Y|R]) :- medlem(X,R).
Singleton variables: [Y]
```

On the right, the execution results are displayed for three queries:

- Query 1:** `medlem("van halen",[1, roxette, 4, "motley crue", 3, "van halen"]).`
 - Singleton variables: [R]
 - Singleton variables: [Y]
 - Result: **true**
 - Buttons: Next, 10, 100, 1,000, Stop
- Query 2:** `member(abba(M1, M2, M3,M4), [roxette(marie, per), van_halen(david, eddie, alex, anthony),abba(agneta, anni-frid, bjorn, benny)]).`
 - Singleton variables: [R]
 - Singleton variables: [Y]
 - Results:
 - M1 = agneta,**
 - M2 = anni-frid,**
 - M3 = bjorn,**
 - M4 = benny**
 - Buttons: Next, 10, 100, 1,000, Stop
- Query 3:** `?- medlem(abba(M1, M2, M3,M4), [roxette(marie, per), van_halen(david, eddie, alex, anthony),abba(agneta, anni-frid, bjorn, benny)]).`

At the bottom right, there are tabs for 'Examples', 'History', and 'Solutions', along with a checkbox for 'table results' and a 'Run!' button.

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).

The screenshot shows the SWISH Prolog IDE interface. The editor on the left contains the following Prolog code:

```
1 medlem(X,[X|R]).  
   Singleton variables: [R]  
2 medlem(X,[Y|R]) :- medlem(X,R).  
   Singleton variables: [Y]  
3 abba(agneta, anni-frid, bjorn, benny).
```

The right-hand pane displays the execution results. The first query, `member(abba(M1, M2, M3,M4), [roxette(marie, per), van_halen(david, eddie, alex, anthony), abba(agneta, anni-frid, bjorn, benny)])`, returns `true` with bindings `M1 = agneta`, `M2 = anni-frid`, `M3 = bjorn`, and `M4 = benny`. The second query, `medlem(abba(M1, M2, M3,M4), [roxette(marie, per), van_halen(david, eddie, alex, anthony), abba(agneta, anni-frid, bjorn, benny)])`, also returns `true` with the same bindings. The third query, `abba(benny, X, Y, Z).`, returns `false`. The bottom of the interface shows tabs for Examples, History, and Solutions, along with a checkbox for 'table results' and a 'Run!' button.

Prolog tries to unify `abba(benny, X, Y, Z)` with `abba(agneta, anni-frid, bjorn, benny)`. Since `benny` \neq `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 $X \geq Y$.
- 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), $X \leq Y$.


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

K

Search


Q


K

021

Program examples movies +

```
1 medlem(X,[X|R]).
  Singleton variables: [R]
2 medlem(X,[Y|R]) :- medlem(X,R).
  Singleton variables: [Y]
3 % For question 5
4 abba(agneta, anni-frid, bjorn, benny).
5
6 % For question 6
7 largest_element(X, [X]).
8 largest_element(X, [X|Rest]) :- largest_element(Y, Rest), X >=
9 largest_element(Y, [X|Rest]) :- largest_element(Y, Rest), Y >
10
11 smallest_element(X, [X]).
12 smallest_element(X, [X|Rest]) :- smallest_element(Y, Rest), X
13 smallest_element(Y, [X|Rest]) :- smallest_element(Y, Rest), Y
14
15 difference(Diff, List) :-
16     largest_element(Max, List),
17     smallest_element(Min, List),
18     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
K
02

Program examples movies +

```

1 medlem(X,[Y|R]) :- medlem(X,R).
Singleton variables: [Y]
2
3 % For question 5
4 abba(agneta, anni-frid, bjorn, benny).
5
6 % For question 6
7 largest_element(X, [X]).
8 largest_element(X, [X|Rest]) :- largest_element(Y, Rest), X
9 largest_element(Y, [X|Rest]) :- largest_element(Y, Rest), Y
10
11 smallest_element(X, [X]).
12 smallest_element(X, [X|Rest]) :- smallest_element(Y, Rest),
13 smallest_element(Y, [X|Rest]) :- smallest_element(Y, Rest),
14
15 difference(Diff, List) :-
16     largest_element(Max, List),
17     smallest_element(Min, List),
18     Diff is Max - Min.
19
20
21 % For question 7
22 % mean/2: calculates the mean (average) of a List
23 mean(List, Mean) :-
24     sum_list(List, Sum),
25     length(List, Len),
26     Len > 0, % avoid division by zero
27     Mean is Sum / Len.
28
29 % concat/3: concatenates two Lists
30 % concat is deprecated, can use atom_concat
31 concat([], L, L).
32 concat([H|T], L2, [H|R]) :- concat(T, L2, R).
33
34
35 analyze_list(List, Length, Sum, Mean, Reversed, Concatenated
36     Length(List, Length),
37     sum_list(List, Sum),
38     mean(List, Mean),
39     reverse(List, Reversed),
40     concat(List, Reversed, Concatenated).
41

```



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(_)).

0 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 .