

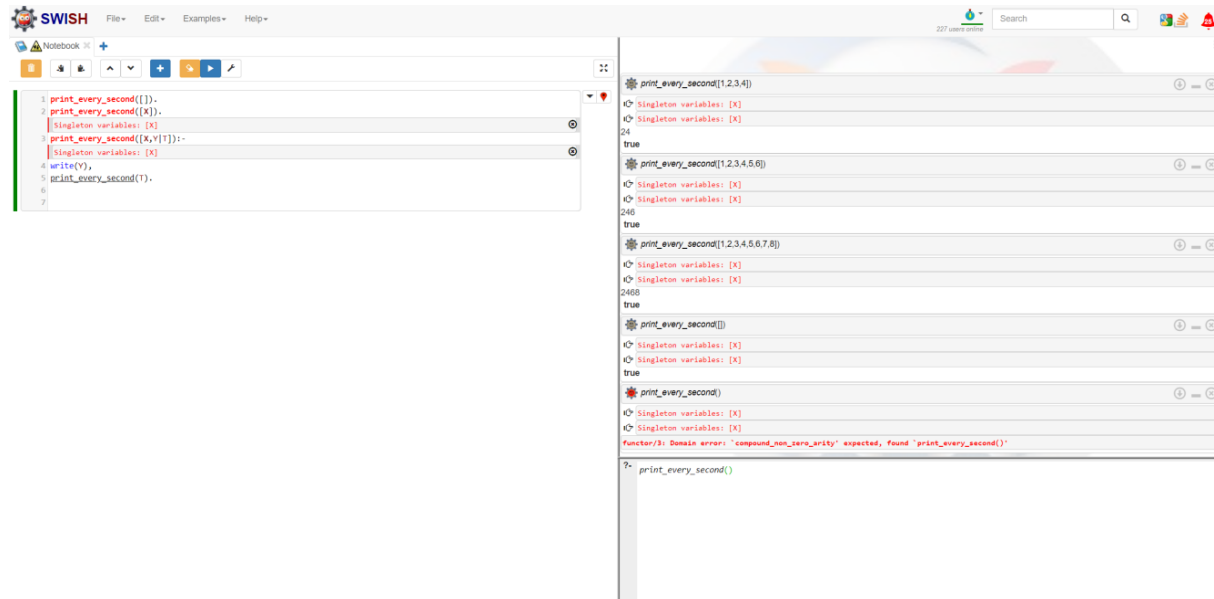
David Alejandro Velázquez Valdéz A01632648

Felix David De Haro Soto A01637589

Diego Ortiz Mariscal A01552000

Actividad 5.1 Programación Lógica

1. Write a predicate print every second/1 to print every other element in a list, beginning at the second element —i.e. the 2nd, 4th, 6th elements etc. It should always succeed provided it is given a list as its argument.

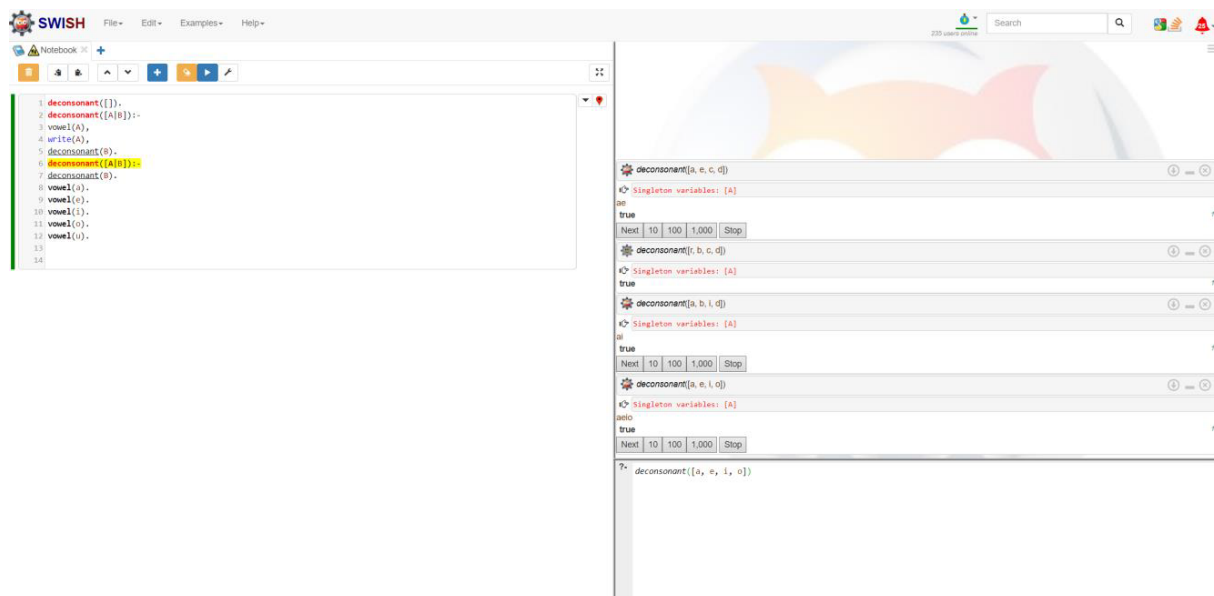


```
1 print_every_second([]).
2 print_every_second([_]).
3 Singleton variables: [X]
4 print_every_second([_,_|_]):-
5 Singleton variables: [X]
6 write(','),
7 print_every_second([_]).
```

Execution results:

- `print_every_second([1,2,3,4])`: true
- `print_every_second([1,2,3,4,5,6])`: true
- `print_every_second([1,2,3,4,5,6,7,8])`: true
- `print_every_second([])`: true
- `print_every_second()`: Domain error: 'compound_non_var_arity' expected, found 'print_every_second()'

2. Write a predicate deconsonant/1 to print any element of a list that isn't a consonant (i.e. we want to print out the vowels fa,e,i,o,u,g). It should always succeed provided it is given a list as its argument (we assume that the input list only contains vowels and consonants).

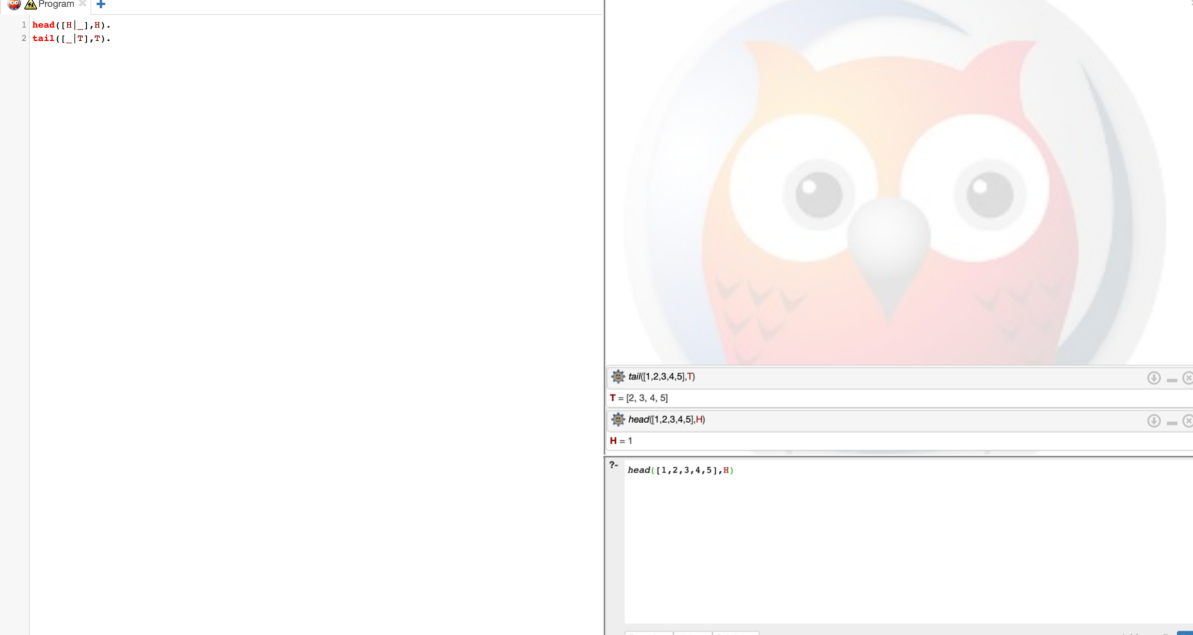


```
1 deconsonant([]).
2 deconsonant([_]).
3 write(A),
4 write(','),
5 deconsonant([_]).
6 deconsonant([_]).
7 deconsonant([_]).
8 vowel(a),
9 vowel(e),
10 vowel(i),
11 vowel(o),
12 vowel(u),
13 vowel(g),
14
```

Execution results:

- `deconsonant([a, e, c, d])`: true
- `deconsonant([c, b, c, d])`: true
- `deconsonant([a, b, l, d])`: true
- `deconsonant([a, e, l, d])`: true

3. Write a predicate `head/2` which takes a list as its first argument and returns the head of the list as its second argument. It should fail if there is no first element.

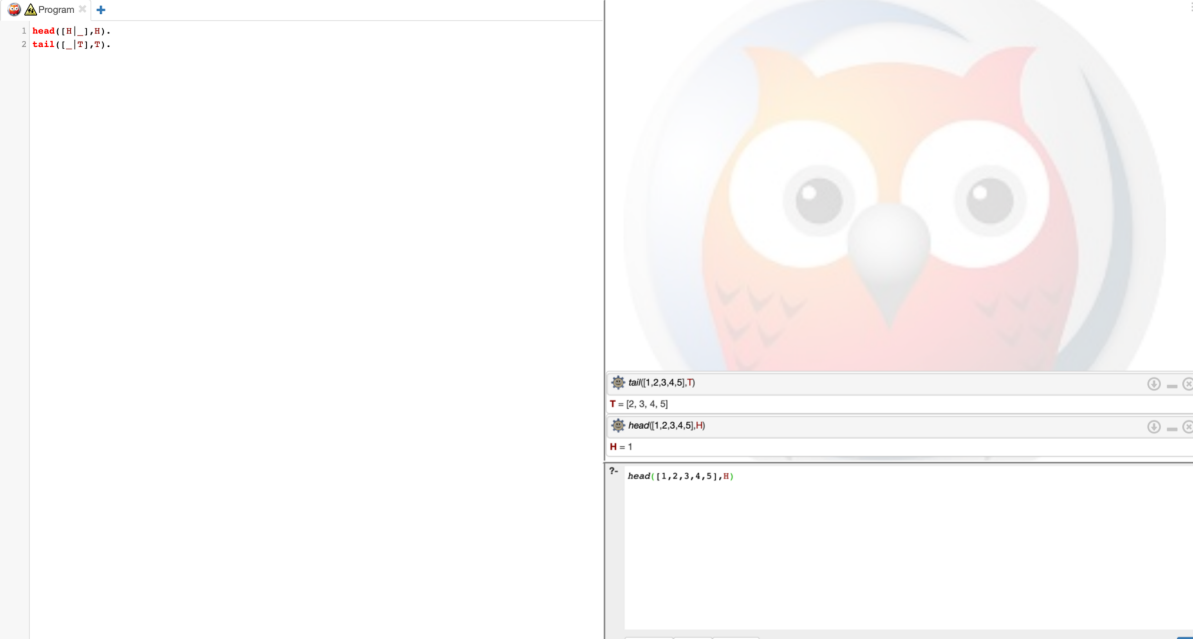


The image shows a Prolog IDE interface. On the left, a code editor contains the following program:

```
1 head([_:_], H).
2 tail([_:_], T).
```

On the right, a query window shows the query `head([1,2,3,4,5], H)` with the variable `H` bound to `1`. The results window shows the query `head([1,2,3,4,5], H)` with the variable `H` bound to `1`.

4. Write a predicate `tail/2` which takes a list as its first argument and returns the tail of the list as its second argument. It should fail if there is no first element.



The image shows a Prolog IDE interface. On the left, a code editor contains the following program:

```
1 head([_:_], H).
2 tail([_:_], T).
```

On the right, a query window shows the query `tail([1,2,3,4,5], T)` with the variable `T` bound to `[2, 3, 4, 5]`. The results window shows the query `tail([1,2,3,4,5], T)` with the variable `T` bound to `[2, 3, 4, 5]`.

7. Write a predicate `fact/2` which takes a natural number as first argument, and returns the factorial of the number.

`fact(0, 1).`

`fact(X, R) :- X > 0, X1 is X-1, fact(X1, R2), R is R2*X.`

The screenshot shows the SWISH Prolog IDE interface. On the left, the program editor contains the following code:

```

1 fact(0, 1).
2 fact(X, R) :- X > 0, X1 is X-1, fact(X1, R2), R is R2*X.

```

On the right, the execution results are displayed, showing the results of the `fact(X, R)` predicate for various values of `X`:

```

?- fact(1, X)
X = 1
false
?- fact(2, X)
X = 2
false
?- fact(3, X)
X = 6
false
?- fact(4, X)
X = 24
false
?- fact(5, X)
X = 120
false

```

Below the results, there is a prompt for a new query: `?- fact(5, X)`.

16. Define `sum/2` to take a list of integers as input and return the output as their sum.

`sum([], 0).`

`sum([Head|Tail], R) :-`

`sum(Tail, SumTail),`

`R is Head + SumTail.`

The screenshot shows the SWISH Prolog IDE interface. On the left, the program editor contains the following code:

```

1 sum([], 0).
2 sum([Head|Tail], R) :-
3   sum(Tail, SumTail),
4   R is Head + SumTail.

```

On the right, the execution results are displayed, showing the results of the `sum(L, R)` predicate for various lists `L`:

```

?- sum([5, 10, 20], X)
X = 35
?- sum([1, 2, 3, 4, 5, 6], X)
X = 21

```

Below the results, there is a prompt for a new query: `?- sum([1, 2, 3, 4, 5, 6], X)`.

19. Write a predicate `split/4` that splits a list into two parts, the length of the first part is given.

`split(L, 0, [], L).`

`split([Head|TailX], N, [Head|TailY], List2) :- N > 0, N1 is N - 1, split(TailX, N1, TailY, List2).`

SWISH

File Edit Examples Help

242 users online

Search

Program

```
1 split(L,0,[],L).
2 split([Head|TailX],N,[Head|TailY],List2) :- N > 0, N1 is N - 1, split(TailX,N1,TailY,List2).
```

split([1,2,3,4],3,X,Y)

X = [1,2,3],
Y = [4]
false

split([1,2,3,4,7,8,9,10],3,X,Y)

X = [1,2,3],
Y = [4,7,8,9,10]
false

split([1,2,3,4,7,8,9,10],6,X,Y)

X = [1,2,3,4,7,8],
Y = [9,10]
false

split([],0,X,Y)

X = Y, Y = []

?- split([],0,X,Y)

Examples History Solutions

☐ table results

Run