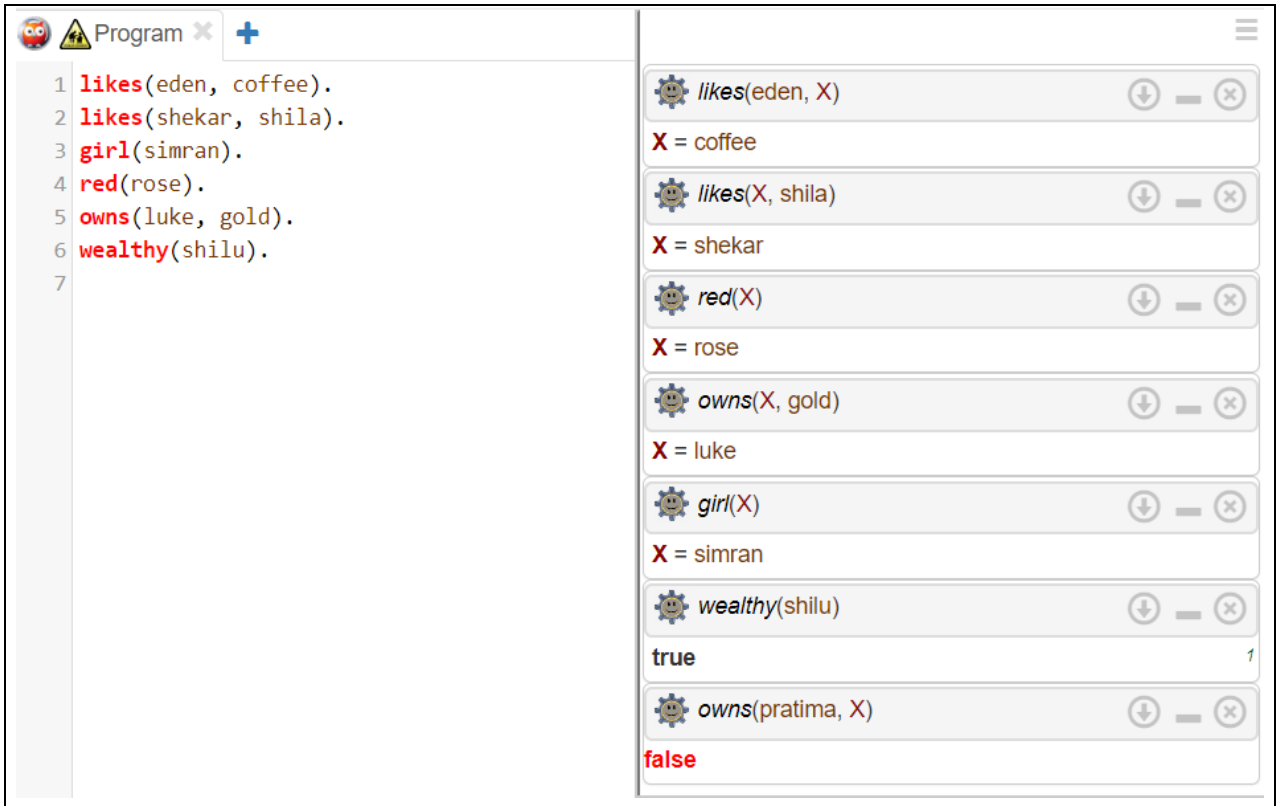
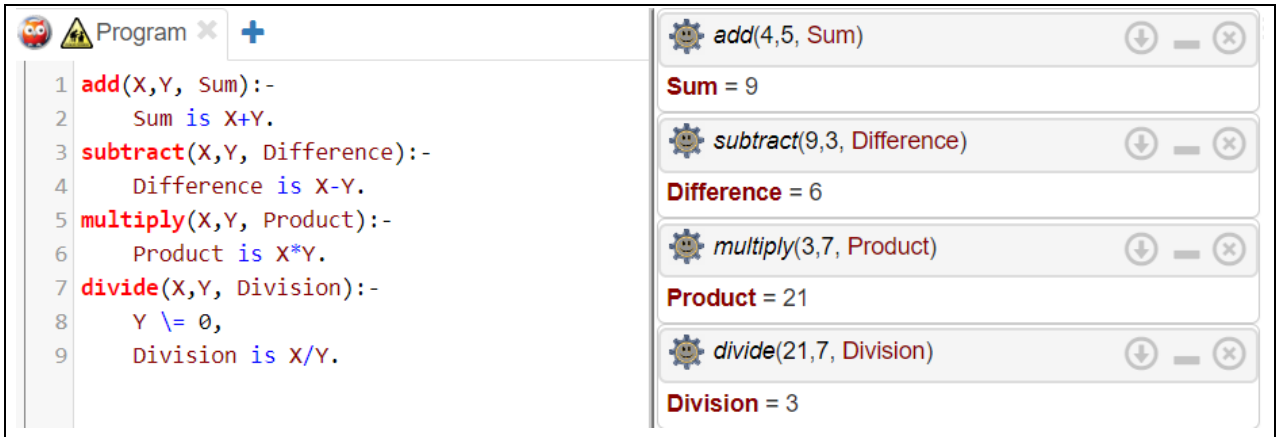
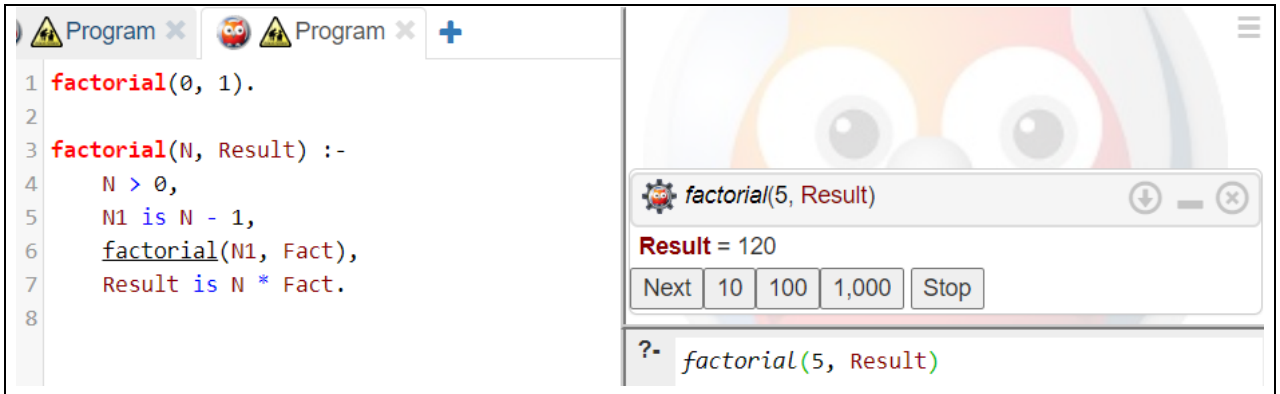

Program
likes(eden, coffee).
likes(shekar, shila).
girl(simran).
red(rose).
owns(luke, gold).
wealthy(shilu).
likes(eden, X)
X = coffee
likes(X, shila)
X = shekar
red(X)
X = rose
owns(X, gold)
X = luke
girl(X)
X = simran
wealthy(shilu)
true
1
owns(pratima, X)
false


Program
1 add(X,Y, Sum):-
2 Sum is X+Y.
3 subtract(X,Y, Difference):-
4 Difference is X-Y.
5 multiply(X,Y, Product):-
6 Product is X*Y.
7 divide(X,Y, Division):-
8 Y \= 0,
9 Division is X/Y.
add(4,5, Sum)
Sum = 9
subtract(9,3, Difference)
Difference = 6
multiply(3,7, Product)
Product = 21
divide(21,7, Division)
Division = 3


Program
Program
1 factorial(0, 1).
2
3 factorial(N, Result) :-
4 N > 0,
5 N1 is N - 1,
6 factorial(N1, Fact),
7 Result is N * Fact.
8
factorial(5, Result)
Result = 120
Next
10
100
1,000
Stop
?-
factorial(5, Result)

Program

```
1  conflict(A, B) :- A = B.
2
3  map_coloring(Region1, Region2, Region3, Regi
4      Colors = [red, green, blue, yellow],
5      member(Region1, Colors),
6      member(Region2, Colors),
7      member(Region3, Colors),
8      member(Region4, Colors),
9      member(Region5, Colors),
10     \+ conflict(Region1, Region2),
11     \+ conflict(Region1, Region5),
12     \+ conflict(Region2, Region3),
13     \+ conflict(Region2, Region4),
14     \+ conflict(Region3, Region4),
15     \+ conflict(Region4, Region5).
16
```

map_coloring(Region1, Region2, Region3, Region4, Region5)

Region1 = Region3, **Region3** = red,
Region2 = Region5, **Region5** = green,
Region4 = blue

Next | 10 | 100 | 1,000 | Stop

?- *map_coloring*(Region1, Region2, Region3, Region4, Region5)

Program

```
1  tower_of_hanoi(1, Source, Destination, _) :-
2      write('Move disk 1 from '), write(Source),
3      write(' to '), write(Destination), nl.
4
5  tower_of_hanoi(N, Source, Destination, Auxiliary) :-
6      N > 1,
7      M is N - 1,
8      tower_of_hanoi(M, Source, Auxiliary, Destination),
9      write('Move disk '), write(N), write(' from '),
10     write(Source),write(' to '), write(Destination),nl,
11     tower_of_hanoi(M, Auxiliary, Destination, Source).
12
```

tower_of_hanoi(3, 'A', 'C', 'B').

```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
true
```

Next | 10 | 100 | 1,000 | Stop

```
?- tower_of_hanoi(3, 'A', 'C', 'B').
```


Program
Program
1 celsius_to_fahrenheit(C, F) :-
2 F is (C * 9 / 5) + 32.
3
4
celsius_to_fahrenheit(32, X)
X = 89.6

Program

```
1  mortal(X) :- human(X).
2  human(socrates).
3  die(X):- mortal(X).
4
5  lives_on_land(X) :-
6      animal(X), \+ (lives_in_water(X)).
7  lives_in_water(X) :- animal(X).
8  lives_in_water(sharks).
9  lives_in_water(X) :- fish(X).
10 animal(X) :- fish(X).
11 fish(whales).
12 fish(sharks).
```

die(socrates)

true 1

lives_in_water(whales)

true 1

lives_on_land(sharks)

false

Program | Program

```
male(bikash).
male(roman).
male(mani).
male(james).
female(ramila).
female(shila).
female(anu).
female(shweta).

parent(bikash, anu).
parent(ramila, anu).
parent(bikash, shweta).
parent(ramila, shweta).
parent(roman, james).
parent(shila, james).
parent(mani, simon).
parent(anu, simon).
parent(james, shekhar).
parent(shweta, shekhar).

mother(X, Y) :- parent(X, Y), female(X).
father(X, Y) :- parent(X, Y), male(X).
grandmother(X, Y) :- parent(X, Z), parent(Z, Y), female(X).
sister(X, Y) :- parent(P, X), parent(P, Y), female(X), X \= Y.
brother(X, Y) :- parent(P, X), parent(P, Y), male(X), X \= Y.
uncle(X, Y) :- parent(P, Y), brother(X, P).

ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

mother(X,shweta)

X = ramila

grandmother(X,shekhar)

X = ramila

X = shila

false

parent(X,james)

X = roman

X = shila

sister(X,shila)

false

uncle(X,shekhar)

false

ancestor(X,simon)

X = mani

X = anu

X = bikash

X = ramila

false

?-