Fr. Conceicao Rodrigues College of Engineering

Department of Computer Engineering

EXPERIMENT 2

Practical No:	2
Title:	Program in prolog to implement simple facts and Queries
Date of Performance:	5-02-2025
Date of Submission:	11-02-2025
Roll No:	9913
Name of the Student:	Mark Lopes

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Theory understanding(02)	02(Correct)	NA	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Output/Test Cases	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Academic Year	2024-25	Estimated Time	Experiment No. 2 – 02 Hours
Course & Semester	T.E. (CE) – Sem. VI	Subject Name	CSC604: Artificial Intelligence
Chapter No.	04	Chapter Title	Knowledge and Reasoning
Experiment Type	Knowledge and Reasoning	Software	Prolog

AIM: Write a program in prolog to implement simple facts and Queries

1. ***Ram likes mango.***
2. ***Seema is a girl.***
3. ***Bill likes Cindy.***
4. ***Rose is red.***
5. ***John owns gold.***

Clauses

```
likes(ram ,mango).
girl(seema). red(rose).
likes(bill ,cindy).
owns(john ,gold).
```

Goal

```
?- likes (ram,What).
What = mango.
1 solution.
```

Assignment:

Aim: Write facts for following:

1. Ram likes apple.
2. Ram is taller than Mohan.
3. My name is Subodh.
4. Apple is fruit.
5. Orange is fruit.
6. Ram is male.

AIM: Write simple queries for following facts.

Simple Queries

Now that we have some facts in our Prolog program, we can consult the program in the listener and query, or call, the facts. This chapter, and the next, will assume the Prolog program contains only facts. Queries against programs with rules will be covered in a later chapter.

Prolog queries work by pattern matching. The query pattern is called a **goal**. If there is

a fact that matches the goal, then the query succeeds and the listener responds with 'yes.' If there is no matching fact, then the query fails and the listener responds with 'no.'

Prolog's pattern matching is called **unification**. In the case where the logic base contains only facts, unification succeeds if the following three conditions hold.

- The predicate named in the goal and logic base are the same.
- Both predicates have the same arity.
- All of the arguments are the same.

Before proceeding, review figure 3.1, which has a listing of the program so far.

The first query we will look at asks if the office is a room in the game. To pose this, we would enter that goal followed by a period at the listener prompt.

?- room(office). yes

Prolog will respond with a 'yes' if a match was found. If we wanted to know if the attic was a room, we would enter that goal.

?- room(attic). no

Solution:-

clauses

likes(ram ,mango).
girl(seema).
red(rose).

likes(bill ,cindy).
owns(john ,gold).

queries

?-likes(ram,What).
What= mango
?-likes(Who,cindy).
Who= cindy

?-red(What).
What= rose
?-owns(Who,What).
Who= john
What= gold

Prolog:

```
female(pam).
female(liz).
female(pat).
female(ann).

male(jim).
male(bob).
male(tom).
male(peter).

parent(pam,bob).
parent(tom,bob).
parent(peter,jim).
parent(bob,pat).
parent(bob,peter).
parent(liz,pat).
parent(liz,peter).

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

Output:

```
PS C:\AI\LAB 2> swipl .\family.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- female(X).
X = pam ;
X = liz ;
X = pat ;
X = ann.

2 ?- male(pat).
false.

3 ?- sister(X,Y).
X = pat,
Y = peter ;
X = pat,
Y = peter ;
false.

4 ?- trace.
true.

[trace] 4 ?- parent(X,Y).
   Call: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(pam, bob) ? creep
X = pam,
Y = bob ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(tom, bob) ? creep
X = tom,
Y = bob ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(peter, jim) ? creep
X = peter,
Y = jim ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(bob, pat) ? creep
X = bob,
Y = pat ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(bob, peter) ? creep
X = bob,
Y = peter ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(liz, pat) ? creep
X = liz,
Y = pat ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(liz, peter) ? creep
```

```
X = liz,
Y = pat ;
   Redo: (12) parent(_2844, _2846) ? creep
   Exit: (12) parent(liz, peter) ? creep
X = liz,
Y = peter.

[trace] 5 ?- wife(X,Y).
   Call: (12) wife(_236, _238) ? creep
   Call: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(pam, bob) ? creep
   Call: (13) parent(_238, bob) ? creep
   Exit: (13) parent(pam, bob) ? creep
   Call: (13) female(pam) ? creep
   Exit: (13) female(pam) ? creep
   Call: (13) male(pam) ? creep
   Fail: (13) male(pam) ? creep
   Redo: (13) parent(_238, bob) ? creep
   Exit: (13) parent(tom, bob) ? creep
   Call: (13) female(pam) ? creep
   Exit: (13) female(pam) ? creep
   Call: (13) male(tom) ? creep
   Exit: (13) male(tom) ? creep
   Exit: (12) wife(pam, tom) ? creep
X = pam,
Y = tom ;
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(tom, bob) ? creep
   Call: (13) parent(_238, bob) ? creep
   Exit: (13) parent(pam, bob) ? creep
   Call: (13) female(tom) ? creep
   Fail: (13) female(tom) ? creep
   Redo: (13) parent(_238, bob) ? creep
   Exit: (13) parent(tom, bob) ? creep
   Call: (13) female(tom) ? creep
   Fail: (13) female(tom) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(peter, jim) ? creep
   Call: (13) parent(_238, jim) ? creep
   Exit: (13) parent(peter, jim) ? creep
   Call: (13) female(peter) ? creep
   Fail: (13) female(peter) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Call: (13) parent(_238, pat) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_238, pat) ? creep
   Exit: (13) parent(liz, pat) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(bob, peter) ? creep
```

```
   Exit: (13) parent(bob, pat) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_238, pat) ? creep
   Exit: (13) parent(liz, pat) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Call: (13) parent(_238, peter) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_238, peter) ? creep
   Exit: (13) parent(liz, peter) ? creep
   Call: (13) female(bob) ? creep
   Fail: (13) female(bob) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(liz, pat) ? creep
   Call: (13) parent(_238, pat) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Call: (13) female(liz) ? creep
   Exit: (13) female(liz) ? creep
   Call: (13) male(bob) ? creep
   Exit: (13) male(bob) ? creep
   Exit: (12) wife(liz, bob) ? creep
X = liz,
Y = bob ;
   Redo: (13) parent(_238, pat) ? creep
   Exit: (13) parent(liz, pat) ? creep
   Call: (13) female(liz) ? creep
   Exit: (13) female(liz) ? creep
   Call: (13) male(liz) ? creep
   Fail: (13) male(liz) ? creep
   Redo: (13) parent(_236, _1568) ? creep
   Exit: (13) parent(liz, peter) ? creep
   Call: (13) parent(_238, peter) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Call: (13) female(liz) ? creep
   Exit: (13) female(liz) ? creep
   Call: (13) male(bob) ? creep
   Exit: (13) male(bob) ? creep
   Exit: (12) wife(liz, bob) ? creep
X = liz,
Y = bob ;
   Redo: (13) parent(_238, peter) ? creep
   Exit: (13) parent(liz, peter) ? creep
   Call: (13) female(liz) ? creep
   Exit: (13) female(liz) ? creep
   Call: (13) male(liz) ? creep
   Fail: (13) male(liz) ? creep
   Fail: (12) wife(_236, _238) ? creep
false.
```

```
[trace] 6 ?-
uncle(X,Z).
   Call: (12) uncle(_236, _238) ? creep
   Call: (13) brother(_236, _1568) ? creep
   Call: (14) parent(_2380, _236) ? creep
   Exit: (14) parent(pam, bob) ? creep
   Call: (14) parent(pam, _1568) ? creep
   Exit: (14) parent(pam, bob) ? creep
   Call: (14) male(bob) ? creep
   Exit: (14) male(bob) ? creep
   Call: (14) bob\==bob ? creep
   Fail: (14) bob\==bob ? creep
   Redo: (14) parent(_2380, _236) ? creep
   Exit: (14) parent(tom, bob) ? creep
   Call: (14) parent(tom, _1568) ? creep
   Exit: (14) parent(tom, bob) ? creep
   Call: (14) male(bob) ? creep
   Exit: (14) male(bob) ? creep
   Call: (14) bob\==bob ? creep
   Fail: (14) bob\==bob ? creep
   Redo: (14) parent(_2380, _236) ? creep
   Exit: (14) parent(peter, jim) ? creep
   Call: (14) parent(peter, _1568) ? creep
   Exit: (14) parent(peter, jim) ? creep
   Call: (14) male(jim) ? creep
   Exit: (14) male(jim) ? creep
   Call: (14) jim\==jim ? creep
   Fail: (14) jim\==jim ? creep
   Redo: (14) parent(_2380, _236) ? creep
   Exit: (14) parent(bob, pat) ? creep
   Call: (14) parent(bob, _1568) ? creep
   Exit: (14) parent(bob, pat) ? creep
   Call: (14) male(pat) ? creep
   Fail: (14) male(pat) ? creep
   Redo: (14) parent(bob, _1568) ? creep
   Exit: (14) parent(bob, peter) ? creep
   Call: (14) male(pat) ? creep
   Fail: (14) male(pat) ? creep
   Redo: (14) parent(_2380, _236) ? creep
   Exit: (14) parent(bob, peter) ? creep
   Call: (14) parent(bob, _1568) ? creep
   Exit: (14) parent(bob, pat) ? creep
   Call: (14) male(peter) ? creep
   Exit: (14) male(peter) ? creep
   Call: (14) peter\==pat ? creep
   Exit: (14) peter\==pat ? creep
   Exit: (13) brother(peter, pat) ? creep
   Call: (13) parent(pat, _238) ? creep
   Fail: (13) parent(pat, _238) ? creep
   Redo: (14) parent(bob, _1568) ? creep
   Exit: (14) parent(bob, peter) ? creep
   Call: (14) male(peter) ? creep
   Exit: (14) male(peter) ? creep
   Call: (14) peter\==peter ? creep
```

```
   Exit: (14) male(peter) ? creep
   Call: (14) peter\==pat ? creep
   Exit: (14) peter\==pat ? creep
   Exit: (13) brother(peter, pat) ? creep
   Call: (13) parent(pat, _238) ? creep
   Fail: (13) parent(pat, _238) ? creep
   Redo: (14) parent(liz, _1568) ? creep
   Exit: (14) parent(liz, peter) ? creep
   Call: (14) male(peter) ? creep
   Exit: (14) male(peter) ? creep
   Call: (14) peter\==peter ? creep
   Fail: (14) peter\==peter ? creep
   Fail: (13) brother(_236, _1568) ? creep
   Fail: (12) uncle(_236, _238) ? creep
false.

[trace] 7 ?-
grandparent(X,Y).
   Call: (12) grandparent(_238, _240) ? creep
   Call: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(pam, bob) ? creep
   Call: (13) parent(bob, _240) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Exit: (12) grandparent(pam, pat) ? creep
X = pam,
Y = pat ;
   Redo: (13) parent(bob, _240) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Exit: (12) grandparent(pam, peter) ? creep
X = pam,
Y = peter ;
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(tom, bob) ? creep
   Call: (13) parent(bob, _240) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Exit: (12) grandparent(tom, pat) ? creep
X = tom,
Y = pat ;
   Redo: (13) parent(bob, _240) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Exit: (12) grandparent(tom, peter) ? creep
X = tom,
Y = peter ;
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(peter, jim) ? creep
   Call: (13) parent(jim, _240) ? creep
   Fail: (13) parent(jim, _240) ? creep
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(bob, pat) ? creep
   Call: (13) parent(pat, _240) ? creep
   Fail: (13) parent(pat, _240) ? creep
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(bob, peter) ? creep
   Call: (13) parent(peter, _240) ? creep
```

```
   Exit: (13) parent(peter, jim) ? creep
   Exit: (12) grandparent(bob, jim) ? creep
X = bob,
Y = jim ;
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(liz, pat) ? creep
   Call: (13) parent(pat, _240) ? creep
   Fail: (13) parent(pat, _240) ? creep
   Redo: (13) parent(_238, _1572) ? creep
   Exit: (13) parent(liz, peter) ? creep
   Call: (13) parent(peter, _240) ? creep
   Exit: (13) parent(peter, jim) ? creep
   Exit: (12) grandparent(liz, jim) ? creep
X = liz,
Y = jim.
```

PostLab Assignment:

Aim: Using the following facts answer the question

1. Find car make that cost is exactly 2,00,000/-
2. Find a car make that costs less than 5 lacs.
3. List all the cars available.
4. Is there any car which costs more than 10 lacs.

Prolog:

```
car(mazda, 200000).
car(toyota, 350000).
car(honda, 150000).
car(bmw, 800000).
car(mercedes, 1200000).
car(ford, 400000).
car(nissan, 450000).
car(audi, 950000).
car(chevrolet, 600000).
car(kia, 300000).
car(volvo, 700000).
car(jaguar, 1300000).
car(subaru, 500000).
car(lexus, 1100000).
car(suzuki, 250000).
car(tesla, 3000000).
car(land_rover, 1500000).
car(ferrari, 25000000).
car(porsche, 1800000).
car(fiat, 180000).

% Rule 1: Find car make that costs exactly 2,00,000
find_2(Make) :- car(Make, 200000).

% Rule 2: Find car make that costs less than 5,00,000
find_5(Make) :- car(Make, Price), Price < 500000.

% Rule 3: List all available cars
list_all_cars :-
    car(Make, Price),
    write(Make), write(' costs '), write(Price), nl,
    fail.
list_all_cars.
% Rule 4: Check if any car costs more than 10 lacs
```

```
car_10_lacs :- car(_, Price), Price > 1000000.
```

Output:

```
PS C:\AI\LAB 2> swipl .\cars.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- car(X,Y).
X = mazda,
Y = 200000 ;
X = toyota,
Y = 350000 ;
X = honda,
Y = 150000 ;
X = bmw,
Y = 800000 ;
X = mercedes,
Y = 1200000 ;
X = ford,
Y = 400000 ;
X = nissan,
Y = 450000 ;
X = audi,
Y = 950000 ;
X = chevrolet,
Y = 600000 ;
X = kia,
Y = 300000 ;
X = volvo,
Y = 700000 ;
X = jaguar,
Y = 1300000 ;
X = subaru,
Y = 500000 ;
X = lexus,
Y = 1100000 ;
X = suzuki,
Y = 250000 ;
X = tesla,
Y = 3000000 ;
X = land_rover,
Y = 1500000 ;
X = ferrari,
Y = 25000000 ;
X = porsche,
Y = 1800000 ;
X = fiat,
Y = 180000.

2 ?- trace.
true.
```

```
[trace] 2 ?- find_2(X).
   Call: (12) find_2(_236) ? creep
   Call: (13) car(_236, 200000) ? creep
   Exit: (13) car(mazda, 200000) ? creep
   Exit: (12) find_2(mazda) ? creep
X = mazda.
```

```
[trace] 3 ?- find_5(X).
   Call: (12) find_5(_6132) ? creep
   Call: (13) car(_6132, _7424) ? creep
   Exit: (13) car(mazda, 200000) ? creep
   Call: (13) 200000<500000 ? creep
   Exit: (13) 200000<500000 ? creep
   Exit: (12) find_5(mazda) ? creep
X = mazda ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(toyota, 350000) ? creep
   Call: (13) 350000<500000 ? creep
   Exit: (13) 350000<500000 ? creep
   Exit: (12) find_5(toyota) ? creep
X = toyota ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(honda, 150000) ? creep
   Call: (13) 150000<500000 ? creep
   Exit: (13) 150000<500000 ? creep
   Exit: (12) find_5(honda) ? creep
X = honda ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(bmw, 800000) ? creep
   Call: (13) 800000<500000 ? creep
   Fail: (13) 800000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(mercedes, 1200000) ? creep
   Call: (13) 1200000<500000 ? creep
   Fail: (13) 1200000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(ford, 400000) ? creep
   Call: (13) 400000<500000 ? creep
   Exit: (13) 400000<500000 ? creep
   Exit: (12) find_5(ford) ? creep
X = ford ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(nissan, 450000) ? creep
   Call: (13) 450000<500000 ? creep
   Exit: (13) 450000<500000 ? creep
   Exit: (12) find_5(nissan) ? creep
X = nissan ;
```

```
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(audi, 950000) ? creep
   Call: (13) 950000<500000 ? creep
   Fail: (13) 950000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(chevrolet, 600000) ? creep
   Call: (13) 600000<500000 ? creep
   Fail: (13) 600000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(kia, 300000) ? creep
   Call: (13) 300000<500000 ? creep
   Exit: (13) 300000<500000 ? creep
   Exit: (12) find_5(kia) ? creep
X = kia ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(volvo, 700000) ? creep
   Call: (13) 700000<500000 ? creep
   Fail: (13) 700000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(jaguar, 1300000) ? creep
   Call: (13) 1300000<500000 ? creep
   Fail: (13) 1300000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(subaru, 500000) ? creep
   Call: (13) 500000<500000 ? creep
   Fail: (13) 500000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(lexus, 1100000) ? creep
   Call: (13) 1100000<500000 ? creep
   Fail: (13) 1100000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(suzuki, 250000) ? creep
   Call: (13) 250000<500000 ? creep
   Exit: (13) 250000<500000 ? creep
   Exit: (12) find_5(suzuki) ? creep
X = suzuki ;
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(tesla, 3000000) ? creep
   Call: (13) 3000000<500000 ? creep
   Fail: (13) 3000000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(land_rover, 1500000) ? creep
   Call: (13) 1500000<500000 ? creep
   Fail: (13) 1500000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(ferrari, 25000000) ? creep
   Call: (13) 25000000<500000 ? creep
   Fail: (13) 25000000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(porsche, 1800000) ? creep
   Call: (13) 1800000<500000 ? creep
   Fail: (13) 1800000<500000 ? creep
   Redo: (13) car(_6132, _7424) ? creep
   Exit: (13) car(fiat, 180000) ? creep
```

```
[trace] 4 ?- list_all_cars.
   Call: (12) list_all_cars ? creep
   Call: (13) car(_1476, _1478) ? creep
   Exit: (13) car(mazda, 200000) ? creep
   Call: (13) write(mazda) ? creep
mazda
   Exit: (13) write(mazda) ? creep
   Call: (13) write(' costs ') ? creep
 costs
   Exit: (13) write(' costs ') ? creep
   Call: (13) write(200000) ? creep
200000
   Exit: (13) write(200000) ? creep
   Call: (13) nl ? creep

   Exit: (13) nl ? creep
   Call: (13) fail ? creep
   Fail: (13) fail ? creep
   Redo: (13) car(_1476, _1478) ? creep
   Exit: (13) car(toyota, 350000) ? creep
   Call: (13) write(toyota) ? creep
toyota
   Exit: (13) write(toyota) ? creep
   Call: (13) write(' costs ') ? creep
 costs
   Exit: (13) write(' costs ') ? creep
   Call: (13) write(350000) ? creep
350000
   Exit: (13) write(350000) ? creep
   Call: (13) nl ? creep

   Exit: (13) nl ? creep
   Call: (13) fail ? creep
   Fail: (13) fail ? creep
   Redo: (13) car(_1476, _1478) ? creep
   Exit: (13) car(honda, 150000) ? creep
   Call: (13) write(honda) ? creep
honda
   Exit: (13) write(honda) ? creep
   Call: (13) write(' costs ') ? creep
 costs
   Exit: (13) write(' costs ') ? creep
   Call: (13) write(150000) ? creep
150000
   Exit: (13) write(150000) ? creep
   Call: (13) nl ? creep

   Exit: (13) nl ? creep
   Call: (13) fail ? creep
   Fail: (13) fail ? creep
   Redo: (13) car(_1476, _1478) ? creep
   Exit: (13) car(bmw, 800000) ? creep
   Call: (13) write(bmw) ? creep
bmw
```

```
car_10_lacs.
   Call: (12) car_10_lacs ? creep
   Call: (13) car(_1550, _1476) ? creep
   Exit: (13) car(mazda, 200000) ? creep
   Call: (13) 200000>1000000 ? creep
   Fail: (13) 200000>1000000 ? creep
   Redo: (13) car(_4790, _1476) ? creep
   Exit: (13) car(toyota, 350000) ? creep
   Call: (13) 350000>1000000 ? creep
   Fail: (13) 350000>1000000 ? creep
   Redo: (13) car(_8030, _1476) ? creep
   Exit: (13) car(honda, 150000) ? creep
   Call: (13) 150000>1000000 ? creep
   Fail: (13) 150000>1000000 ? creep
   Redo: (13) car(_11270, _1476) ? creep
   Exit: (13) car(bmw, 800000) ? creep
   Call: (13) 800000>1000000 ? creep
   Fail: (13) 800000>1000000 ? creep
   Redo: (13) car(_14510, _1476) ? creep
   Exit: (13) car(mercedes, 1200000) ? creep
   Call: (13) 1200000>1000000 ? creep
   Exit: (13) 1200000>1000000 ? creep
   Exit: (12) car_10_lacs ? creep
true .
```