Artificial Intelligence Laboratory

[CS 4271]

Assignment 1

Prolog Programs

Submitted by

Sriparno Ganguly

Enrol No. – 2020CSB004

Question 1: Find the Last Element of a List

Code:

```
% Base Case :
last_element(X, [X]).
% Recursive rule:
last_element(X, [_|Tail]):-
    last_element(X, Tail).
```

Sample Output:

```
?- last_element(Last, [1, 2, 3, 4, 5]).
Last = 5.
```

Question 2: Append Two Lists

Code:

```
% Base case:
append_lists([], L, L).
% Recursive rule:
append_lists([X|Xs], Y, [X|Z]) :-
    append_lists(Xs, Y, Z).
```

Sample Output:

```
?- append_lists([1, 2, 3], [4, 5, 6], Result).
Result = [1, 2, 3, 4, 5, 6].
```

Question 3: Reverse a List

Code:

```
% Base case:
reverse_list([], []).
% Recursive rule:
reverse_list([X|Xs], Reversed) :-
    reverse_list(Xs, ReversedTail),
    append_lists(ReversedTail, [X], Reversed).
```

Sample Output:

```
?- reverse_list([1, 2, 3, 4, 5], Reversed).
Reversed = [5, 4, 3, 2, 1].
```

Question 4: Check Whether a List is a Palindrome

Code:

```
% Base case: An empty list is a palindrome.
palindrome([]).
% Base case: A list with a single element is a palindrome.
palindrome([_]).
% Recursive rule:
palindrome([X|Xs]) :-
    append_lists(Inner, [X], Xs),
    reverse_list(Inner, ReversedInner),
    append_lists([X], ReversedInner, [X|Xs]).
```

Sample Output:

```
?- palindrome([1, 2, 3, 2, 1]).
true.
```

Question 5: Find the Kth Element of a List

Code:

```
% Base case:
element_at(X, [X|_], 1).
% Recursive rule:
element_at(X, [_|Xs], K) :-
    K > 1,
    K1 is K - 1,
    element_at(X, Xs, K1).
```

Sample Output:

```
?- element_at(Elem, [1, 2, 3, 4, 5], 3).
Elem = 3.
```

Question 6: Sum and Average of a List

Code:

```
% Base case:
sum_and_avg([], 0, 0).
% Recursive case:
sum_and_avg([H|T], Sum, Avg) :-
    sum_and_avg(T, TailSum, TailAvg),
    Sum is H + TailSum,
    Avg is Sum / (1 + TailAvg).
```

Sample Output:

```
?- sum_and_avg([1, 2, 3, 4, 5], Sum, Avg).
Sum = 15,
Avg = 3.
```

Question 7: GCD of Two Numbers

Code:

```
% Base case:
gcd(X, 0, X).
gcd(0, Y, Y).
% Recursive rule:
gcd(X, Y, Result) :-
    Y > 0,
    Z is X mod Y,
    gcd(Y, Z, Result).
```

Sample Output:

```
?- gcd(48, 18, G).
G = 6.
```

Question 8: Prime or Not Prime

Code:

```
% Base case: 0 and 1 are not prime.
is_prime(0) :- false.
is_prime(1) :- false.
% Predicate to check if N is prime.
is_prime(N) :-
    N > 1,
    is_prime_helper(N, 2).
```

Sample Output:

```
?- is_prime(13).
true.
```

Question 9: Prime Factors

Code:

```
% Define a predicate to find the prime factors of a number.
prime_factors(N, Factors) :-
    N > 0,
    prime_factors_helper(N, 2, Factors).
```

Sample Output:

```
?- prime_factors(24, Factors).
Factors = [2, 2, 2, 3].
```

Question 10: Goldberg's Conjecture

Code:

```
% Predicate to find two prime numbers that sum up to a given even integer
goldbach(N, [P1, P2]) :-
    N > 2,
    N mod 2 =:= 0,
    goldbach_helper(N, 3, P1),
    P2 is N - P1,
    is_prime(P2).
```

Sample Output:

```
?- goldbach(28, Pair).
Pair = [5, 23].
```

Question 11: Fibonacci Numbers

Code:

```
% Base case: The first Fibonacci number is 0.
fibonacci(0, 0).
% Base case: The second Fibonacci number is 1.
fibonacci(1, 1).
% Recursive rule: Calculate the Nth Fibonacci number.
fibonacci(N, Result) :-
    N > 1,
    N1 is N - 1,
    N2 is N - 2,
    fibonacci(N1, Fib1),
    fibonacci(N2, Fib2),
    Result is Fib1 + Fib2.
```

Sample Output:

```
?- fibonacci(6, Fib).
Fib = 8.
```

Question 12: Database of Facts

A | Uncle Relationship

Code:

```
% Rules for defining aunts and uncles
aunt(Aunt, NieceNephew) :-
    parent(Parent, NieceNephew),
    sibling(Aunt, Parent),
    female(Aunt).

uncle(Uncle, NieceNephew) :-
    parent(Parent, NieceNephew),
    sibling(Uncle, Parent),
    male(Uncle).
```

Sample Outputs:

```
?- uncle(keith, ann).
Yes
?- uncle(ann, mary).
No
?- uncle(keith, X).
X = ann ;
No
?- uncle(john, ann).
No
?- uncle(X, Y).
X = keith,
Y = ann ;
No
```

B | Half-Sister Relationship

Code:

```
% Predicate to check if two people share at least one parent
halfsister(X, Y) :-
    parent(Z, X),
    parent(Z, Y),
    X \= Y,
    \+ sibling(X, Y).
```

Sample Outputs:

```
?- halfsister(ann, sylvia).
Yes
?- halfsister(X, sylvia).
X = ann ;
No
?- halfsister(X, Y).
X = ann,
Y = sylvia ;
X = sylvia,
Y = ann ;
No
```