**Example 1: Basic Syntax and Input/Output**

**Example Program**:

% Hello World program in Prolog

hello\_world :-

write('Hello, World!').

greet :-

write('What is your name? '),

read(Name),

format('Hello, ~w!', [Name]).

**Practice Exercises**:

1. Modify the **hello\_world** program to greet the user with "Goodbye!" instead.
2. Enhance the **greet** program to ask for the user's age and respond accordingly.

**Example 2: Basic Operations**

**Example Program**:

% Arithmetic operations

add(X, Y, Result) :- Result is X + Y.

subtract(X, Y, Result) :- Result is X - Y.

% Factorial calculation

factorial(0, 1).

factorial(N, Result) :-

N > 0,

N1 is N - 1,

factorial(N1, SubResult),

Result is N \* SubResult.

**Practice Exercises**:

1. Implement a Prolog predicate to calculate the square of a number.
2. Write a predicate to compute the nth Fibonacci number.

**Example 3: Conditional Statements**

**Example Program**:

% If-Else Statements

is\_even(X) :-

0 is X mod 2,

write('Even').

is\_even(X) :-

1 is X mod 2,

write('Odd').

% Comparison Operators

is\_greater(X, Y) :-

X > Y,

write('X is greater than Y').

is\_greater(X, Y) :-

X =< Y,

write('X is not greater than Y').

**Practice Exercises**:

1. Create a Prolog predicate to determine if a number is positive, negative, or zero.
2. Write a program to check if a given year is a leap year or not.

**Example 4: Lists and Cut Operator**

**Example Program**:

% List Operations

list\_length([], 0).

list\_length([\_|T], Len) :-

list\_length(T, Len1),

Len is Len1 + 1.

% Using Cut Operator

is\_positive(X) :-

X > 0,

!.

is\_positive(X) :-

X =< 0.

**Practice Exercises**:

1. Implement a predicate to find the maximum element in a list.
2. Write a Prolog rule to remove duplicates from a list.

**Example 5: Debugging**

**Example Program (with errors)**:

% Debugging Exercise

greater\_than\_ten(X) :-

X > 10,

write('Greater than 10').

greater\_than\_ten(X) :-

X =< 10,

write('Not greater than 10').

**Practice Exercises**:

1. Identify and correct the errors in the **greater\_than\_ten** predicate.
2. Debug a faulty program that is intended to find the sum of elements in a list.

These examples and practice exercises cover a range of topics in Prolog programming, providing hands-on experience with basic syntax, operations, conditional statements, lists, cut operator, and debugging techniques.

Family Tree problems:

% Facts

male(john).

male(peter).

male(mike).

female(mary).

female(susan).

female(linda).

parent(john, peter).

parent(john, mary).

parent(mary, susan).

parent(peter, mike).

parent(linda, mike).

% Rules

father(F, C) :-

male(F),

parent(F, C).

mother(M, C) :-

female(M),

parent(M, C).

sibling(X, Y) :-

parent(Z, X),

parent(Z, Y),

X \= Y.

**Practice Exercises**:

1. Add more relationships to the family tree (e.g., grandparent, aunt, uncle).
2. Write a predicate to find all the children of a given parent.
3. Extend the family tree to include great-grandparent relationships.
4. Implement a predicate to check if two individuals are cousins.
5. Modify the family tree to include spouses and write rules to determine marital relationships.

**Query Exercises**:

* Find all children of John: ?- parent(john, Child).
* List all grandchildren of John: ?- parent(john, Child), parent(Child, Grandchild).
* Find all siblings of Mary: ?- sibling(mary, Sibling).
* Who are the mothers in the family? ?- mother(Mother, \_).
* List all uncles in the family: ?- sibling(Uncle, Parent), male(Uncle).
* Find all cousins of Mike: ?- parent(Parent1, Mike), sibling(Parent1, Parent2), parent(Parent2, Cousin), Cousin \= Mike.
* Who are the grandparents in the family? ?- parent(Parent, Person), parent(Parent, ParentOfParent), Person \= ParentOfParent.
* List all females who have children: ?- mother(Female, \_).
* Find all siblings and their children: ?- sibling(X, Y), parent(X, Child), format('~w is a sibling of ~w and has child ~w.~n', [X, Y, Child]), fail.
* List all males in the family with their children: ?- male(Male), parent(Male, Child), format('~w has child ~w.~n', [Male, Child]), fail.

These queries will allow you to explore the relationships defined in the family tree example and practice querying in Prolog. Feel free to modify and extend the queries as per your learning objectives.