SDE 1: Prolog - Family Relationship

In this assignment, you will use Prolog to model a family tree and define relationships such as parent, sibling, cousin, and grandparent. You will use facts and rules to represent these relationships and allow Prolog to infer new relationships through queries.

Submission Instructions:

Please submit the following items on Canvas:

1. The Prolog source file (.pl) containing your defined facts and rules (including advanced predicates).
2. A text file containing the queries you wrote to test the relationships.
3. A screenshot showing the output of your queries when executed in Prolog.

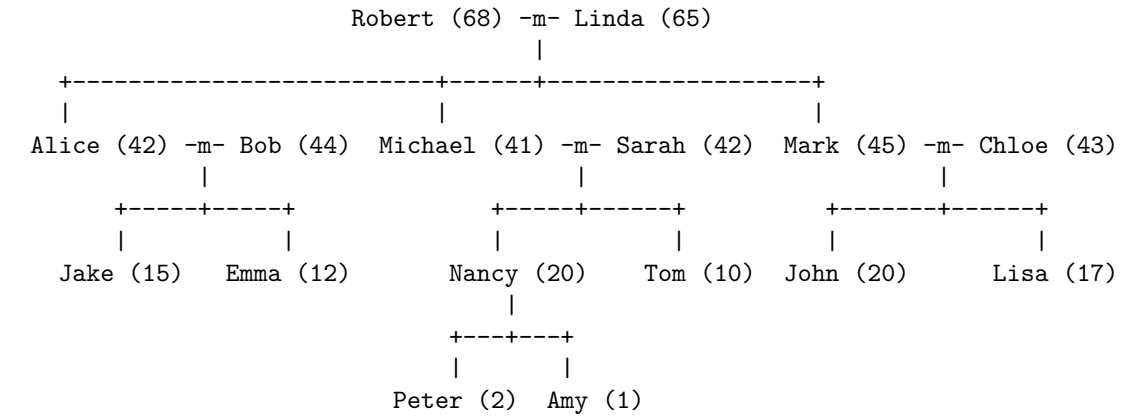
Objectives:

- Define basic family relationships using facts.
- Write rules to infer more complex relationships.
- Use Prolog's query mechanism to test your rules and answer questions about family relationships.

Problem Description:

You are provided with a family tree. You are asked to represent the tree in Prolog using facts and define rules to deduce more complex relationships. The family members' ages are included, and you need to define rules that use these ages to infer who is the "older" sibling.

Family Tree:



The family tree is structured as follows, where "-m-" denotes a marital relationship.

- Robert and Linda are married.
- They have three children: Alice, Michael, and Mark.
- Alice is married to Bob, and they have two children: Jake and Emma.
- Michael is married to Sarah, and they have two children: Nancy and Tom. Nancy has two children: Peter and Amy.
- Mark is married to Chloe, and they have two children: John and Lisa.
- Family members' ages are included.

You are welcome to expand the given family tree by adding additional members; however, please ensure that no existing members are removed or altered.

Exercises

Exercise 1: Defining Facts

Create Prolog facts to represent the following:

1. (5 pts) parent(Parent, Child) - A fact that represents a parent-child relationship.
2. (2 pts) married(Person1, Person2) - A fact to represent a marriage.
3. (2 pts) male(Person) - A fact to indicate that a person is male.
4. (2 pts) female(Person) - A fact to indicate that a person is female.
5. (4 pts) age(Person, Age) - A fact to represent the age of a person.

Exercise 2: Defining Rules

Write rules to define complex relationships. Use recursion and Prolog's declarative approach to represent these relationships.

1. (5 pts) Sibling Relationship:

   Define a rule `sibling(Person1, Person2)` to determine if two people are siblings. Siblings share at least one parent.

2. (5 pts) Grandparent Relationship:

   Define a rule `grandparent(Grandparent, Grandchild)` that identifies grandparents.

3. (5 pts) Cousin Relationship:

   Define a rule `cousin(Person1, Person2)` for cousin relationships. Cousins have parents who are siblings.

4. (5 pts) Uncle/Aunt Relationship:

   Define rules `uncle(Uncle, Child)` and `aunt(Aunt, Child)` to infer whether someone is an uncle or aunt. Uncles and aunts are the siblings of a parent.

5. (5 pts) Ancestor Relationship:

Define a *recursive* rule `ancestor(Ancestor, Descendant)` to determine if a person is an ancestor of another.

Exercise 3: Defining Rules on Age-Based Sibling Relationships

Use the age/2 fact to write rules that compare ages and deduce relationships based on who is older or younger among siblings.

1. (10pts) Older Sibling Relationship:

   Define a rule older_sibling(Older, Younger) to determine if one sibling is older than another.

2. (10pts) Bigger Brother/Bigger Sister:

   Define rules bigger_brother(Brother, Sibling) and bigger_sister(Sister, Sibling) that combine gender and age to infer whether a brother or sister is older than a sibling.

Exercise 3: Queries

Once you've defined the facts and rules, write Prolog queries to answer the following questions about the family tree:

1. (2 pt) Who are the siblings of Alice?
2. (2 pt) Who are the cousins of Jake?
3. (2 pt) Who are the grandparents of Peter?
4. (2 pt) Who are the uncles of Emma?
5. (2 pt) Is Robert an ancestor of Amy?
6. (2 pt) Who is the aunt of Peter?
7. (2 pt) List all descendants of Robert.
8. (2 pt) Find all people who are both a parent and a grandparent.
9. (2 pt) Who is the bigger brother of Emma?
10. (2 pt) Who is the older sibling between Nancy and Tom?

Exercise 4: Advanced predicates

Implement the following predicates and test them with appropriate queries.

1. (5 pts) Brother-in-law/Sister-in-law Relationship:

   Please implement predicates `brother_in_law(BrotherInLaw, Person)` and `sister_in_law(SisterInLaw, Person)` to represent relationships where someone is a sibling of a spouse or married to a sibling.

2. (5 pts) "nth" Ancestor of a Person:

   Please implement a predciat to find an ancestor at a specific generation level, where n is the number of generations between the person and their ancestor (e.g., the nth ancestor of someone could be their parent, grandparent, great-grandparent, etc.).

Hint: You may write a recursion program.

```
nth_ancestor(N, Ancestor, Person).
```

Example Query:

```
?- nth_ancestor(2, Ancestor, tom).
```

Expected Output: Based on the family tree, Tom's 2nd ancestor (grandparent) is:

```
Ancestor = robert ;
Ancestor = linda ;
```

3. (5 pts) Common Ancestor:

   Define a rule common_ancestor(Person1, Person2, Ancestor) that identifies the common ancestor between two people.

   Example Query:

   ```
   ?- common_ancestor(peter, emma, Ancestor).
   ```

   Expected Output:

   ```
   Ancestor = robert ;
   Ancestor = linda.
   ```

4. (5 pts) Common Ancestor List:

   Create a predicate common_ancestors(Person1, Person2, AncestorList) that returns a list of all common ancestors between two people.

   Hint: You may want to use a predicate function ***`findall`***.

   Example Query:

   ```
   ?- common_ancestors(peter, emma, Ancestors).
   ```

   Expected Output:

   ```
   Ancestors = [robert, linda].
   ```