Prolog
Let’s use OOP to simulate something.
A person has a name.
A person can be a safety officer.
A person can have at most one boss.

We can access a person's name.
We can make someone be a safety officer.
We can determine whether a person is a safety officer.
We can access a person's boss.

Maria is Jeff’s boss.
Jeff is Melissa's boss.
Jeff is Kerry’s boss.
Melissa is Ben’s boss.
Melissa is Colleen’s boss.
Kerry is Lelia’s boss.
Kerry is Adam’s boss.
Lelia is a safety officer.
A person has a name.

```java
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    // getters...
    // auto-generated hashCode and equals...
}

Person ben = new Person("Ben");
System.out.println(ben.getName());
```
A person can be a safety officer.

```java
Person lelia = new Person("Lelia");
lelia.makeSafetyOfficer();
```

```java
public class Person {
    ...
    private boolean isSafetyOfficer;
    ...
    public Person(String name) {
        this.name = name;
        this.isSafetyOfficer = false;
    }
    ...
    public void makeSafetyOfficer() {
        this.isSafetyOfficer = true;
    }
    ...
}
```
A person can have at most one boss.

```java
Person jeff = new Person("Jeff", maria);
System.out.println(jeff.getBoss().getName());
System.out.println(maria.getBoss());
```

```java
public class Person {
    // ...
    private Person boss;
    // ...

    public Person(String name) {
        this.name = name;
        this.isSafetyOfficer = false;
        this.boss = null;
    }

    public Person(String name, Person boss) {
        this(name);
        this.boss = boss;
    }
    // ...

    public Person getBoss() {
        return this.boss;
    }
    // ...
}
```
A person has a name.
A person can be a safety officer.
A person can have at most one boss.

We can access a person's name.
We can make someone be a safety officer.
We can determine whether a person is a safety officer.
We can access a person's boss.

Does A work with B? (i.e., are A and B coworkers?)
A and B are coworkers if and only if they have the same boss.

Is B the employee of A?
B is the employee of A if and only if A is B’s boss.

Who are A's employees?
A’s employees are all the people whose boss is A.

Who are A's safety officers?
A’s safety officers are any coworkers of A who are safety officers.
Does $A$ work with $B$?

$A$ works with $B$ if and only if they have the same boss.

```java
def worksWith(Person person):
    myBoss = this.getBoss()
    theirBoss = person.getBoss()
    return myBoss != null and myBoss.equals(theirBoss)
```

```
ben.worksWith(colleen)
ben.worksWith(lelia)
```
Is $B$ the employee of $A$?

$B$ is the employee of $A$ if and only if $A$ is $B$'s boss.

```java
public boolean isEmployeeOf(Person person) {
    return person.equals(this.getBoss());
}
```

ben.isEmployeeOf(melissa)
ben.isEmployeeOf(jeff)
Who are A’s employees?

A’s employees are all the people whose boss is A.

```java
kerry.getEmployees()
```

```java
public Collection<Person> getEmployees() {
    ...
}
```
Who are A’s employees?

A’s employees are all the people whose boss is A.

```java
kerry.getEmployees()

private Collection<Person> employees =
    new HashSet<Person>();

public Person(String name, Person boss) {
    this(name);
    this.boss = boss;
    boss.addEmployee(this);
}
```
Who is A’s safety officer?

A’s safety officer is any coworker of A who is a safety officer.

```java
public Person getSafetyOfficer() {
    Person myBoss = this.getBoss();
    if (myBoss == null) {
        return null;
    } else {
        Optional<Person> safetyOfficer =
            myBoss.getEmployees()
                .stream()
                .filter(p -> p.isSafetyOfficer)
                .findFirst();
        if (safetyOfficer.isPresent()) {
            return safetyOfficer.get();
        } else {
            return null;
        }
    }
}
```
Why can’t we just say this?

Maria is Jeff’s boss.
Jeff is Melissa's boss.
Jeff is Kerry’s boss.
Melissa is Ben’s boss.
Melissa is Colleen’s boss.
Kerry is Lelia’s boss.
Kerry is Adam’s boss.
Lelia is a safety officer.
Person A is an employee of person B if B is A's boss.
Person A works with person B if they have the same boss.
Person A’s safety officer is anyone A works with who is a safety officer.
Behold: Prolog!

boss(maria, jeff).
boss(jeff, melissa).
boss(jeff, kerry).
boss(melissa, ben).
boss(melissa, colleen).
boss(kerry, lelia).
boss(kerry, adam).
safetyOfficer(lelia).

employee(PersonA, PersonB) :- boss(PersonB, PersonA).

worksWith(PersonA, PersonB) :-
    boss(Boss, PersonA), boss(Boss, PersonB).

safetyOfficer(PersonA, PersonB) :-
    worksWith(PersonA, PersonB), safetyOfficer(PersonA).
Mudd Mad Libs!

Groups of eight.

Each group member needs:

- a survey
- a pencil
- a piece of paper

Each group needs:

- an envelope of envelopes

Groups of eight.

My name is: ____________________________

first name, last initial

I live in: _____________________________

dorm

I have a: ______________________________

pull a pet from the envelope

I love: ________________________________

pull a sport from the envelope

On Monday, I: _________________________

sport1

On Tuesday, I: _________________________

sport2

On Wednesday, I: ______________________

sport3

On Thursday, I: _________________________

sport4

On Friday, I: __________________________

sport5

On Saturday, I: _________________________

sport6

On Sunday, I: _________________________

sport7

Each group member needs:

- a survey
- a pencil
- a piece of paper

Each group needs:

- an envelope of envelopes
ben

atom

begins w/ lower case
〈name〉 lives in 〈dorm〉.
livesIn(ben, off_campus).

?- livesIn(ben, off_campus).
true.

?- livesIn(ben, zoo).
false.
__________ has a ⟨pet⟩.
__________ has a ⟨pet⟩.

?- hasPet(X, liger).
X = ben ;
X = alex.

?- hasPet(X, unicorn).
X = ashley.
__________ lives in ⟨dorm⟩, loves ⟨sport⟩.
lives in ⟨dorm⟩, loves ⟨sport⟩.

?- livesIn(X, west), lovesSport(X, swimming).
X = alex.

?- livesIn(X, west), lovesSport(X, badminton).
false.
___________ does ⟨sport⟩ on __________.
__________ does ⟨sport⟩ on __________.

?- doesOnDay(Who, teaching, When).
Who = ben,
When = tuesday ;
Who = ben,
When = thursday.
A person is safe if they do not have a(n) ⟨pet⟩.
A person is safe if they do not have a(n) ⟨pet⟩.

\[
\text{safe}(X) :- \text{hasPet}(X, Y), Y \not= \text{liger}.
\]

“head is true, if body is true”
___________ is safe.
_________ is safe.

?- safe(alex).
false.

?- safe(ashley).
true.
A person is happy on a day if that’s the day they do something they love.
A person is happy on a day if that’s the day they do something they love.

happy(X, D) :- doesOnDay(X, S, D), lovesSport(X, S).
__________ is happy today.
_________ is happy today.

?- happy(Who, tuesday).
Who = ben.
__________ will be happy this weekend.
will be happy this weekend.

?- happy(Who, saturday); happy(Who, sunday).
Who = alex;
Who = ashley.