Exercise 3.

?- time = money.
This does not unify because time and money are two different atoms.

?- money = Time.
This unifies with Time = money. Variables in Prolog start with an uppercase letter or underscore, so Time can be instantiated to money.

?- Time = Money.
This unifies with Time = Money, where Time and Money can both be instantiated to the same term.

?- smoothie(X, Y, Z) = smoothie(banana, strawberry, "almond milk").
This unifies with X = banana, Y = strawberry, Z = "almond milk".

?- name(X, "de Vries") = name(sharon, X).
This does not unify because X cannot be both "de Vries" and sharon at the same time.

?- name(X, vries, de) = name(sharon, Y).
This does not unify because the number and arrangement of arguments are different. name/3 cannot unify with name/2.

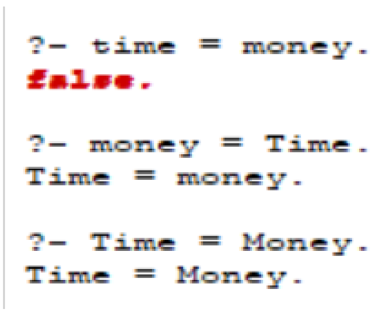
?- pancake(ingredient(flour), topping(syrup)) = pancake(X, Y).
This unifies with X = ingredient(flour), Y = topping(syrup).

?- pancake(ingredient(flour), Y, topping(syrup)) = pancake(ingredient(milk), ingredient(flour), X).
This does not unify because flour and milk are different atoms in ingredient(flour) and ingredient(milk).

?- X = f(Y).
This unifies with X = f(Y). There is no instantiation, since Y remains a variable.

?- f(g(X), a) = f(g(Y), B).
This unifies with X = Y, B = a.

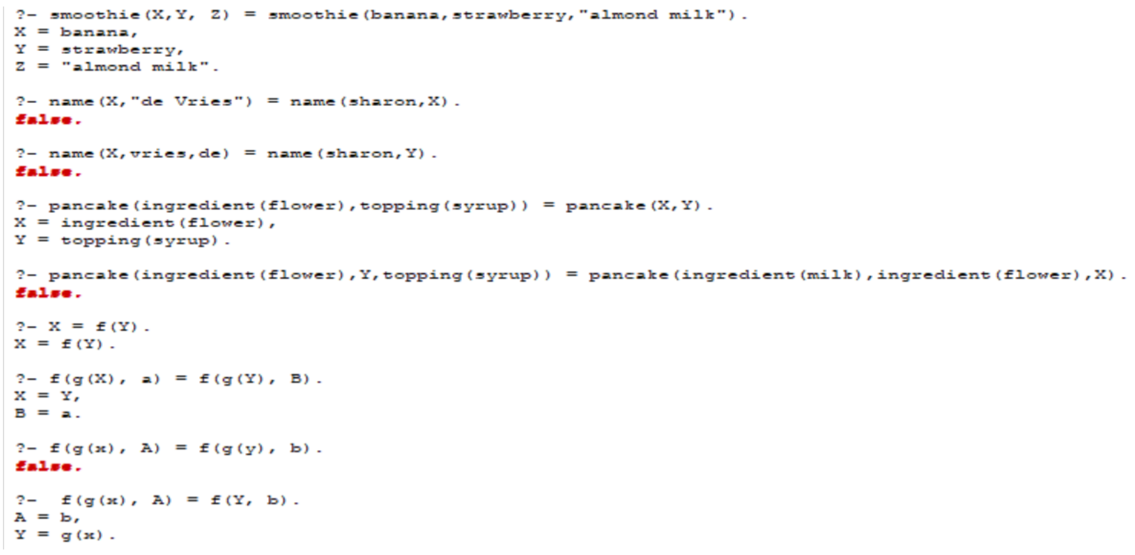
?- f(g(x), A) = f(g(y), b).
This does not unify because the x and the y in g(x) and g(y) are different atoms.



```
?- time = money.
false.

?- money = Time.
Time = money.

?- Time = Money.
Time = Money.
```



```
?- smoothie(X,Y, Z) = smoothie(banana,strawberry,"almond milk").
X = banana,
Y = strawberry,
Z = "almond milk".

?- name(X,"de Vries") = name(sharon,X).
false.

?- name(X,vries,de) = name(sharon,Y).
false.

?- pancake(ingredient(flower),topping(syrup)) = pancake(X,Y).
X = ingredient(flower),
Y = topping(syrup).

?- pancake(ingredient(flower),Y,topping(syrup)) = pancake(ingredient(milk),ingredient(flower),X).
false.

?- X = f(Y).
X = f(Y).

?- f(g(X), a) = f(g(Y), B).
X = Y,
B = a.

?- f(g(x), A) = f(g(y), b).
false.

?-  f(g(x), A) = f(Y, b).
A = b,
Y = g(x).
```

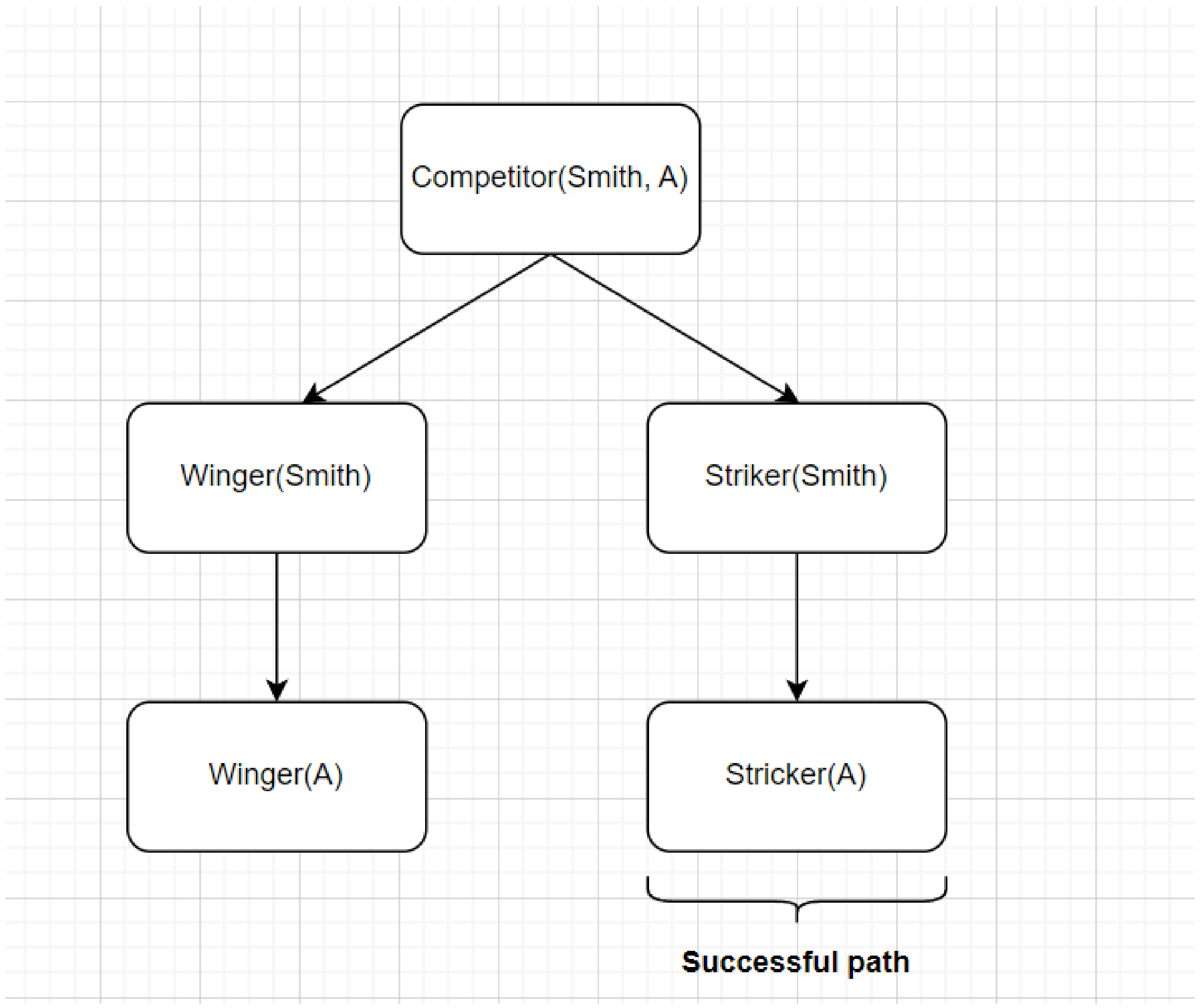
Exercise 4.

1. A call like competitor(Smith, A) will fail, because in Prolog the not(...) function works as a filter, and should go at the end of the query, not in front. It should first find all the names that associate with the variable A, and then filter out those that are not Smith.
2. striker(smith).
   striker(sanchez).
   winger(smith).
   competitor(X, Y) :- (winger(X), winger(Y), not(Y = X)).
   competitor(X, Y) :- (striker(X), striker(Y), not(Y = X)).

   ?- competitor(Smith, A)
   A = sanchez.

   By changing the order of the query, now it first finds all competitors to Smith, and then filters out those who are the same making A equal to sanches.

3.



4. striker(smith).
   striker(sanchez).
   competitor(X, Y) :- (striker(X), striker(Y), not(Y = X)).
   winger(smith).
   competitor(X, Y) :- (winger(X), winger(Y);() not(Y = X)).

Exercise 5

warm(a).
happy(b).
warm(Y):- warm(Y).
summer(X):- warm(Y).
summer(X):- happy(X).

This order will return an infinite number of true values when summer(a). is called. It returns true since summer(a) will give us warm(a). Then warm(a) is plugged into the fourth line in the database which returns itself and gives us the infinite loop.

Exercise 6.

1.

```
subtract(X, 0, X).
subtract(succ(X), Y, succ(Z)) :- subtract(X, Y, Z)
```

2.

```
subtract(0, X, -X).
subtract(succ(X), succ(Y), Z) :- subtract(X, Y, Z).
```

3.

```
subtract(X, 0, X).
subtract(succ(X), succ(Y), Z) :- subtract(X, Y, Z).
subtract(-(X), Y, -(Z)) :- add(X, Y, Z).
subtract(X, -(Y), Z) :- add(X, Y, Z).
subtract(-(X), -(Y), Z) :- subtract(Y, X, Z).

add(0, X, X).
add(succ(X), Y, succ(Z)) :- add(X, Y, Z).
```

4.

```
add(0, Y, Y).
add(X, 0, X).

add(succ(X), Y, succ(Z)):- add(X, Y, Z).
add(-(succ(X)), -(Y), -(succ(Z))):- add(X, Y, Z).
add(succ(X), -(Y), succ(Z)):- subtract(X,Y,Z).
add(-(succ(X)), Y, succ(Z)):- subtract(Y,X,Z).
```