Assignment 1

Question 1

Found in Employee.java and Person.java

Question 2

Found in Driver.java

Question 3

1. food(bread, X) = Food(Y, soup)

Error;

Capital “F” Food is not a valid functor.

2. Bread = soup

Unify;

Bread = soup.

3. Bread = Soup

Unify;

4. food(bread, X, milk) = food(Y, salad, X)

Does Not Unify;

X can’t be both milk and salad.

5. manager(X) = Y

Unify;

The entire thing will be unified with Y.

6. meal(healthyFood(bread), drink(milk)) = meal(X,Y)

Unify;

X= healthyFood(bread)

Y = drink(milk)

7. meal(eat(Z), drink(milk)) = [X]

Does Not Unify;

LHS isn’t a list

8. [eat(Z), drink(milk)] = [X, Y | Z]

Unify;

X = eat(Z) = eat([])

Y = drink(milk)

Z = []

9. f(X, t(b, c)) = f(l, t(Z, c))

Unify;

X=l

Z=b

10. ancestor(french(jean), B) = ancestor(A, scottish(joe))

Unify;

A = french(jean)

B = scottish(joe)

11. meal(healthyFood(bread), Y) = meal(X, drink(water))

Unify;

X = healthyFood(bread);

Y = drink(water)

12. [H|T] = [a, b, c]

Unify;

H = a

T = [b,c]

13. [H, T] = [a, b, c]

Does Not Unify

LHS has 2 terms, RHS has 3 terms

14. breakfast(healthyFood(bread), egg, milk) = breakfast(healthyFood(Y), Y, Z)

Does Not Unify.

Y cannot be both bread and egg

15. dinner(X, Y, Time) = dinner(jack, cook(egg, oil), Evening)

Unify;

X = jack

Y = cook(egg,oil)

Time = Evening

16. k(s(g), Y) = k(X, t(k))

Unify

X = s(g)

Y = t(k)

17. equation(Z, f(x, 17, M), L*M, 17) = equation(C, f(D, D, y), C, E)

Does Not Unify

D cannot be both x and 17

18. a(X, b(c, d), [H|T]) = a(X, b(c, X), b)

Does Not Unify

b is not a list, so cannot unify with [H|T]

Question 4

1. ? field(hit_transfer,engineering).

Ground;

field(hit_transfer,engineering) = field(X,Y) : - course (X,Z), field(Z,Y).

X = hit_transfer; Y = engineering

course(hit_transfer, Z) = course(hit_transfer, mechanical).

Z = mechanical

field(mechanical, engineering) = field(mechanical, engineering).

True.

2. ? lab_number(fine_arts,X).

Non-ground;

lab_number(fine_arts,X) = lab_number(fine_arts, 10).

X = 10.

3. ? field(computer, literature).

Ground;

field(computer,literature) = field(X,Y) : - course (X,Z), field(Z,Y).

X = computer ; Y = literatire

course(computer,Z) does not unify.

False.

4. ? course(X,Y).

Non-ground.

course(X,Y) = course(hit_transfer, mechanical).

X = hit_transfer;

Y = mechanical.

5. ? student(adrian).

Ground.

student(adrian) = student(X) :- student(X,_).

X = adrian.

student(adrian,_) = student(adrian, web_design).

True.

6. ? student(anna, engineering).

Ground.

student(anna,engineering) = student(X,Y) :- field(Z,Y), student(X,Z).

X = anna; Y = engineering.

…

field(Z,engineering) = field(X,Y) :- course(X,Z), field(Z,Y).

…

Z = hit_transfer

student(anna,hit_transfer) = student(anna, hit_transfer).

True.

7. ? student(X, engineering).

Non-ground;

student(X,engineering) = student(X,Y):- field(Z,Y), student(X,Z).

Y = engineering; X = X;

field(Z,engineering) = … = field(mechanical,engineering) = … = field(hit_transfer, engineering).

Z = hit_transfer;

student(X, hit_transfer) = student(anna, hit_transfer).

X = anna;

8. ? student(X,fine-arts), course(fine_arts, Y).

Non-ground;

student(X,fine-arts) = student(X, Y) :- field(Z,Y), student(X,Z).

Y = fine-arts;

field(Z,fine-arts) = field(X,Y):- course(X,Z), field(Z,Y).

…

field(Z, fine-arts) = field(Z, Y) => fails

False; (No matches found).

9. ? field(_,X).

Non-ground;

field(_,X) = field(mechanical, engineering).

X = engineering.

10. ? lab_number(_,X), field(X,Y).

Non-ground;

lab_number(_,X) = lab_number(mechanical, 15) => False

lab_number(_,X) = lab_number(fine_arts, 10) => False

lab_number(_,X) = lab_number(X,Z) => False

False; (No matches found).

11. ? lab_number(X,15), field(X,Y).

Non-ground;

lab_number(X,15) = lab_number(mechanical, 15).

X = mechanical;

field(mechanica,Y) = field(mechanical, engineering).

Y = engineering;

Output:

X = mechanical.

Y = engineering.

12. ? student(X), !, student(X,_). % note to cut here

Non-ground

student(X) = student(X) :- student(X,_)

student(X,_) = student(anna, hit_transfer).

X = anna;

!

Output:

X = anna;

13. ? student(X), student(X,_), !.

student(X) = student(X) :- student(X,_)

student(X,_) = student(anna, hit_transfer).

X = anna;

student(anna,_) = student(anna, hit_transfer).

!

Output :

X = anna ;

14. ? course(X,_), \+ student(_,X). % \+ is for negation (not)

non-ground

course(X,_) = course(hit_transfer,mechanica).

X = hit_transfer;

student(_,hit_transfer) = student(anna, hit_transfer) => True

\+ True => False

…

X = web_design; (True for adrian) => False

X = design_methods; (True for ava) => False

X = poetry; (True for jack) => False

X = leadership; (True for lee) => False

X = biology;

student(_,biology) = student(X,Y) :- field(Z,Y), student(X,Z).

Y = biology;

field(Z,biology) = field(X,Y) : - course(X,Z), field(Z,Y).

…

student(_,biology) => false

\+ false => True.

Output:

X = biology.

Question 5

1) database is in question5.pl

2) ?- student(Person,ID), findall(C,takes(ID,C),List).

3) ?- findall(X,student(X,_),L),length(L,N).

4) ?- findall(X, takes(_,X),Temp), list_to_set(Temp,List).

5) ?- findall(X, takes(_,X),Temp), list_to_set(Temp,List), sort(List,Sorted).

6)

A = comp349

B = comp352

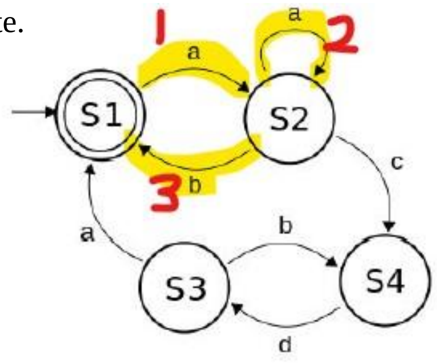
C = [comp361, encs282, engr371, engr391, engr392, mast218, phys284, soen287, soen341]

Question 6

1) database/rules in question6.pl

2) ?- accept([a,a,b]).

3) We follow the path and see that it results in a final state.



Question 7

1) database/rules in question7.pl

2) ?- circuit(0,1,0,1,L).

Question 8

Can be found in question8.pl:

second_half(List,Output):- length(List,N), Half is N/2, half(List, Half, Output).

half(List,N,List):- length(List,Length), Length=< N.

half([_|T],N,List):-length([_|T],Length), Length > N, half(T,N,List).

Question 9

Can’t come up with a query, so used a rule instead. Found in question9.

```
lucas_num(1,[2]).

lucas_num(2,[2,1]).

lucas_num(N,Out):- M is N-1, lucas_num(M,Temp1), last_two(Temp1,A,B),
   C is A + B, append(Temp1,[C],Out).


last_two([A,B],A,B).

last_two([_|T],A,B):- last_two(T,A,B).
```