Allan Paul Lopez, 40000954

William Trembly, 40174212

Rosalie Yelle, 40174964

Assignment 1 – PROLOG part

Q3. Unification:

1. Error: functor Food(Y,soup) cannot start with an upper case letter.
2. Yes: Bread = soup.
3. Yes: For any case where Bread and Soup holds the same value.
4. No because in LHS, X should be salad, but on RHS, X should be milk.
5. No because the arity on LHS is not the same as the arity on RHS
6. Yes: X = healthyFood(bread) and Y = drink(milk)
7. No because you can not unify a query into a list.
8. Error: The list on the RHS has a misuse of "I" operator
9. Can be unified: X = l and Z =b
10. Yes: A = french(jean) and B = scottish(joe)
11. Yes: Y = drink(water) and X = healthyFood(bread)
12. Yes: H = a and T = [b, c]
13. No because the two lists are not of the same size.
14. Yes: Only if healthyFood(egg) unifies with healthyFood(bread). Then, Y=egg and Z = milk
15. Yes: X= jack, Y = cook(egg,oil), Time = Evening)
16. Yes: X = s(g) and Y = t(k)
17. Yes: Only if f(x,17,M) = f(x,x,M) = f(17,17,M), then Z = C, D = 17 or D = x, C = L*y, E = 17
18. No: b is not a list, so it cannot be unified with [H|T]

Q4.

1. **? field(heat_transfer,engineering).** : Ground
    a. The engine will try to match the query, one clause at a time starting from the top and going to the bottom until it is able to unify with the rule field(X,Y).
    b. Prolog will instantiate X to heat_transfer and Y to engineering.
    c. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(heat_transfer,engineering):-course(heat_transfer, Z), field(Z,engineering).
    d. The two new goals will be processed, and the engine will try to unify them for a certain value of Z. In this case, Prolog will find that the two goals are validated with Z = mechanical.
    e. The engine will therefore return **true**.

2. **? lab_number(fine_arts,X).**: Non-Ground

a. The engine will try to find the instantiation for X, if there is any, that is going to make the query true.
b. It will start from the top and make its way to the bottom of the clause given until it can unify the statement with the given condition.
c. If it finds a fact, which is going to be the case, it will try to unify it with it.
d. It will be successful when it arrives to: **lab_number(fine_arts,10).**
e. The engine will therefore return **X = 10.**

3. **? field(computer, literature).** : Ground
    a. The engine will try to match the query, one clause at a time starting from the top and going to the bottom until it is able to unify with the rule field(X,Y).
    b. Prolog will instantiate X to computer and Y to literature.
    c. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(computer,literature):-course(computer, Z), field(Z,literature).
    d. The two new goals will be processed and the engine will try to unify them for a certain value of Z. In this case, Prolog will fail to unify both goals.
    e. The engine will therefore return **false**.

4. **? course(X,Y). :** Non-Ground
    a. The engine will try to find the instantiation for X and Y, if there is any, that is going to make the query true.
    b. It will start from the top and make its way to the bottom of the clause given until it can unify the statement with the given conditions.
    c. If it finds a fact, which is going to be the case, it will try to unify it with it.
    d. It will be successful when it arrives to: **course(heat_transfer, mechanical).**
    e. The engine will therefore return **X=heat_transfer and Y= mechanical**.
    f. If you try to find other answers the engine will do the same steps and find: X= web_design and Y = computer, then X = design_methods and Y = fine-arts, and finally X = poetry and Y = literature.

5. **? student(jeff).** : Ground
    a. The engine will try to match the query one clause at a time starting from the top and going to the bottom until it is able to unify with the rule student(X).
    b. Prolog will instantiate X to jeff.
    c. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(john):- student (john,_).
    d. The new goal will be processed, and the engine will try to unify it with another clause. In this case, Prolog will find that the goal unifies with the rule student(X, Y).
    e. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(john, _):- field (Z, _), student(X, Z).
    f. The two new goals will be processed, and the engine will try to unify them for a certain value of Z. In this case, Prolog will find that the two goals are validated with Z = heat_transfer. (The procedure for field(Z, _) is described in 1.)

g. The engine will therefore return **true**.

6. **? student(john, engineering). :** Ground
    a. The engine will try to match the query going down from the top of the list of clauses given.
    b. It is going to find a rule: student(X, Y) :- field(Z, Y), student(X, Z).
    c. Then, the query will unify with the head of it, instantiating john to X and Y engineering to Y.
    d. Resolution will after apply the substitution of the variables, creating the new rule: student(john, engineering) :- field(Z, engineering), student(john, Z).
    e. For the head to be true, the two goals of the body of the clause must be true. Prolog will evaluate the two goals.
    f. It will try to unify each one, trying to find an instantiation that will work for both goals.
    g. For field(Z, engineering) it will unify with field(mechanical, engineering) instantiating Z with mechanical.
    h. This will instantiate student(john, Z) too with Z = mechanical which will then fail.
    i. It will try to do the same thing with Z=computer and will fail.
    j. The engine will then go to the rule field(X, Y) :- course(X, Z), field(Z, Y). It will be able to unify the two roles: course(heat_transfer, mechanical) and field(mechanical, engineering).
    k. That will verify the second goal of the rule student(john, engineering) :- field(Z, engineering), student(john, Z). Student(john, Z) will become student(john,heat_tranfer) unifying it with the same clause in the data base.
    l. The engine **returns true.**

7. **? student(X, engineering).** : Non-Ground
    a. The engine will try to match the query one clause at a time starting from the top and going to the bottom until it is able to unify with the rule student(X,Y).
    b. Prolog will instantiate Y to engineering.
    c. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(X, engineering):- :- field (Z, engineering), student(X, Z).
    d. The first goal will be processed, and the engine will try to unify it for any value of Z possible. In this case, Prolog will find that the goals are unified when Z = mechanical or Z = computer.
    e. The engine will then try to unify the second goal with Z = mechanical and find any values of X that validates the goal. In this case, no matching X will be found, so the goal will fail for this particular instance of Z.

f. Now, the engine will try to unify the same goal with Z = computer and find any values of X that validates the goal. As in the previous step, no matching X will be found, so the goal will fail for all cases.
g. Therefore, the engine will return **false.**

8. **? student(X, fine-arts), course(fine_arts, Y).:** Non-Ground
    a. The engine will try to match the query, student(X, fine-arts) going down from the top of the list of clauses given.
    b. It will unify with the rule student(X, Y), instantiating Y with fine-arts. Resolution will then apply the substitution of the variables and create: student(X, fine-arts) :- field(Z, fine-arts), student(X, Z).
    c. The engine will then try to unify each goal, starting with field(Z, fine-arts) and then student(X, Z). The engine will not be able to find a match returning false.
    d. The engine will then **return false**.

9. **? field(_, X).** : Non-Ground
    a. The engine will try to match the query, one clause at a time starting from the top and going to the bottom. Since the first argument can be matched to anything, Prolog will return any value of X for which the query unifies. In this case, any value.
    b. The engine will return: **X=engineering; X=engineering; X=art; X=social; X=business;**
    c. Prolog will then unify with the rule field(Y, Z), intantiating Y = _ and Z = X.
    d. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(_,X):-course(_, A), field(A,X).
    e. The two new goals will be processed and the engine will try to unify them for any value of X. The values that were returned previously are the same as the ones which validates the new goals with the exception of business.
    f. Therefore, the engine will return **X=engineering; X=engineering; X=art; X=social; false**.

10. **? lab_number(_, X), field(X, Y). :** Non-Ground
    a. The engine will try to match the query, lab_number(_, X**)** going down from the top of the list of clauses given. Since the first argument can be matched to anything, Prolog will return any value of X for which the query unifies.
    b. The engine will return **X = 15**.
    c. Then the engine is going to then instantiate X to 15 and try to unify the second clause of the query. It will not be able to do it as a number is given.

d. The engine will then **return false**.

11. **? lab_number(X, 15), field(X, Y).** : Non-Ground
    a. The engine will try to match the first clause of the query, one clause at a time starting from the top and going to the bottom of the data base until it is able to unify with the term lab_number(mechanical, 15).
    b. Prolog will then instantiate X to mechanical and try to unify the second clause of the query. In this case, the engine will unify it with field(mechanical, engineer).
    c. Thus, the engine will return: **X = mechanical Y = engineer;**
    d. The engine will continue to unify the first clause of the query with one from the data base. It will be able to unify with the rule lab_number(X,Z).
    e. Prolog will instantiate Z to 15.
    f. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: lab_number(X,15):- course(X,Y), lab_number(Y,15)
    g. The two new goals will be processed, and the engine will try to unify them for a certain value of X and Y. In this case, X = heat_transfer and Y = mechanical.
    h. Prolog will unify the second clause of the query with X = heat_transfer. It will succeed to unify with field(heat_transfer, engineering) via the rule field(X,Y) as shown in 1.
    i. Therefore, the engine will return: **X = heat_transfer Y = engineer; false.**

12. **? student(X), !, student(X,_). % note to cut here :** Non-Ground
    a. The engine will first try to match the query, student(X) going down from the top of the list of clauses given.
    b. It will find the rule: student(X):-student(X,_). Then the it will try to unify the goal of the rule with student(john, heat_transfer).
    c. The unification of the clause is a success and X=john.
    d. The second clause of the query tells us to not backtrack at this point of the search returning **X = john**.

13. **? student(X), student(X,_), !.** : Non-Ground
    a. The engine will try to match the first clause of the query, one clause at a time starting from the top and going to the bottom of the data base until it is able to unify with the rule student(X):-student(X,_)
    b. The goal student(X,_) will be able to unify with student(john, heat_transfer). Therefore, the unification of the first clause of the query will be a success, so X will be instantiated with john.
    c. The second clause of the query student(john,_) will now be verified, and will unify once again with student(john, heat_transfer).
    d. The third statement of the query will be verified. Since it only consists of the cut operator, Prolog will end the unification process and will return: **X=john**.

14. **? course(X, _), \+ student(_,X). % \+ is for negation (not) :** Non-Ground
    a. The engine will first try to match the query, course(X,_) going down from the top of the list of clauses given. Since the second argument can be matched to anything, Prolog will return any value of X for which the query unifies.
    b. The engine will then return **X = heat_transfer; X = web_design; X = design_methods; X = poetry**.
    c. The second clause of the query, \+ student(_,X) will then be evaluated. Where X will be instantiated with first heat_transfer. Since the query is negative and there's student(john, heat_transfer), the engine will **return false**.
    d. It is the same with the other Xs, making the engine **return false**.