HOME WORK ASSIGNMENT – 2
GAURAV NANDA (gaurav324@gmail.com)

QUESTION :- 1

dbase(entryStations, [entryStation, atm, cashierStation, consortium, branch, user]).

% **Table definitions**:
table(entryStation, [entryStationId, stationId, isOperating]).
table(atm, [entryStationId, stationId, isOperating, cashOnHand, dispensed, consortiumId]).
table(cashierStation, [entryStationId, stationId, isOperating, branchId]).
table(consortium, [consortiumId]).
table(branch, [branchId, connected]).
table(user, [userId]).

% **Constraint - 1**
% Maintain the inheritance relation. This would ensure that if someone wants to query an instance of ATM or Cashier Station from Parent Table(i.e. Entry Station), it should return true.

entryStation(A,B,C) :- atm(A,B,C,_,_,_).
entryStation(A,B,C) :- cashierStation(A,B,C,_).

% **Constraint - 2**
% A primary key in a table should always be unique.

primary_key_not_unique :- consortium(A), aggregate_all(count, consortium(A), COUNT), COUNT \= 1, write('Consortium Primary Key is not unique').

primary_key_not_unique :- entryStation(A,_,_), aggregate_all(count, entryStation(A,_,_), COUNT), COUNT \= 1, write('EntryStation primary key is not unique.').

primary_key_not_unique :- branch(A,_), aggregate_all(count, branch(A,_), COUNT), COUNT \= 1, write('Branch primary key is not unique').

primary_key_not_unique :- user(A), aggregate_all(count, user(A), COUNT), COUNT \= 1, write('User Primary key is not unique.').

NOTE: Using constraints 1 and 2, we can ensure that all the EntryStationId's are unique across all the tables.

% **Constraint - 3**
% EntryStationID should not be null in any of the EntryStation, ATM and CashierStation table.

entry_station_id_null :- entryStation(null,_,_), write('Entry Station Id is null.').

% **Constraint - 4**
% Consortium Id should not be null in Consortium and ATM table. We don't need to explicitly check in ATM table, because there is another foreign key constraint that we have put.

consortium_id_null :- consortium(null), write('Consortium Id is null.').

% **Constraint - 5**
% Every Consortium Id present inside ATM table must also be present in Consortium table.

consortium_foreign_key_breaks :- atm(_,_,_,_,_,F), not(consortium(F)), write('Foreign key constraint violation for consortium id in ATM table.').

% **Constraint - 6**
% Branch Id should not be null in Branch and CashierStation table.

branch_id_null :- branch(null, _), write('Branch Id is null.').

% **Constraint - 7**
% Every Branch Id present inside CashierStation table must also be present in Branch table.

branch_foreign_key_breaks :- cashierStation(_,_,_,F), not(branch(F, _)), write('Foreign Key constraint violation for branch id in CashierStation Table.').

% Total Constraints check. This does not check the Inheritance relation.

***q1_all_valid_constraints** :- not(entry_station_id_null), not(consortium_id_null), not(consortium_foreign_key_breaks), not(branch_id_null), not(branch_foreign_key_breaks), not(primary_key_not_unique).*

QUESTION :- 2

dbase(part2, [a, b]).

% **Table definitions**:
table(a, [aId, f, bId]).
table(b, [bId, h]).

% **Constraint - 1**
% aID and bID, which are primary keys should not be null.

```
primary_key_null :- a(null,_,_), write('Primary key of A is null').
primary_key_null :- b(null,_), write('Primary key of B is null').

% Constraint - 2
% There should not be any duplicate aId and bId in A and B table respectively.

q2_primary_key_not_unique :- a(A,_,_), aggregate_all(count, a(A,_,_), COUNT), COUNT \= 1, write('aID is not unique in A table.').
q2_primary_key_not_unique :- b(A,_), aggregate_all(count, b(A,_), COUNT), COUNT \= 1, write('bID is not unique in B table.').

% Constraint – 3
% bID present in A table, must be present in B table (Foreign key constraint.) or it could be null.

foreign_key_breaks :- a(_,_,F), F \= null, not(b(F,_)), write('Foreign key constraint violation.').

q2_all_valid_constraints :- not(primary_key_null), not(q2_primary_key_not_unique), not(foreign_key_breaks).
```