Artificial Intelligence

Nehal Jhajharia (U20CS093)
Practical Examination

Q1)
Domains
Disease -> indication = symbol Patient -> name = string
Patient(P_Id,name,address(building_name,city,zipcode),[treatment(doctor_c ode, disease]).
Predicates
symptom(name,indication) hypothesis(name,disease) response(char)
Where,
● indication = fever, rash, headache, runny_nose, conjunctivitis, cough,
● name= Patient's namebody_ache, chills, sore_throat, sneezing
Disease are as follow:
1. Flu if patient has fever, headache, body_ache, conjunctivitis, chills, sore_throat, runny_nose, cough
2. Common cold if patient has headache, sneezing, sore_throat, runny_nose, chills
3. Chicken pox if patient has fever, chills, body_ache, rash
4. Measles if patient has cough, sneezing, runny_nose
Clauses
For 3 patients, enter symptoms. Make sure every patient should be having more than one disease for data purposes.
Enter that 3 patients details (P_Id,Name, Address, treatment)
Example:
symptom(Patient’s name,fever) :-
write(\"Does \",Patient,\" have a fever (y/n) ?\"), response(Reply),
Reply=\'y\'.
10
Find the results for following questions using PROLOG program:
1. Find the total number of diseases for each patient.
2. Find the name and zip code of each patient.
3. Write P_Id and name of all patients staying in Delhi.

4. List name of all patients treated by doctor D1.
5. List roll no. of all patients suffering from Common cold
6. List building_name and city_code for all patients in the given format (format: [(building_name, citycode)]).
7. List all doctors for each given patient.

```
% Domain declarations
domain(disease, [flu, common_cold, chicken_pox, measles]).
domain(indication, [fever, rash, headache, runny_nose, conjunctivitis, cough, body_ache, chills, sore_throat, sneezing]).
domain(symbol, [patient]).
domain(string, [name]).
domain(char, [y,n]).
domain(city_code, [delhi, mumbai, kolkata, chennai, bangalore]).
domain(doctor_code, [d1, d2, d3, d4, d5]).
domain(zipcode, [111111, 222222, 333333, 444444, 555555]).

% Patient database
patient(1, 'John', address('ABC Apartments', delhi, 111111), [treatment(d1, flu), treatment(d2, common_cold)]).
patient(2, 'Mary', address('XYZ Society', mumbai, 222222), [treatment(d1, chicken_pox), treatment(d3, measles)]).
patient(3, 'Peter', address('PQR Towers', delhi, 333333), [treatment(d4, common_cold), treatment(d5, flu)]).

% Symptoms database
symptom('John', fever).
symptom('John', headache).
symptom('John', body_ache).
symptom('John', conjunctivitis).
symptom('John', chills).
symptom('John', sore_throat).
symptom('John', runny_nose).
symptom('John', cough).
symptom('Mary', fever).
```

```
symptom('Mary', chills).
symptom('Mary', body_ache).
symptom('Mary', rash).
symptom('Mary', sneezing).
symptom('Mary', sore_throat).
symptom('Mary', runny_nose).
symptom('Mary', cough).
symptom('Peter', headache).
symptom('Peter', sneezing).
symptom('Peter', sore_throat).
symptom('Peter', runny_nose).
symptom('Peter', chills).
symptom('Peter', fever).
symptom('Peter', cough).

% Hypotheses database
hypothesis(flu, [fever, headache, body_ache, conjunctivitis, chills, sore_throat,
runny_nose, cough]).
hypothesis(common_cold, [headache, sneezing, sore_throat, runny_nose, chills]).
hypothesis(chicken_pox, [fever, chills, body_ache, rash]).
hypothesis(measles, [cough, sneezing, runny_nose]).

% Question 1:
diseases_count(Patient, Count) :-
        findall(Disease, hypothesis(Patient, Disease), Diseases),
   length(Diseases, Count).
% Question 2:
patient_address(Patient, Name, ZipCode) :-
        patient(Patient, Name, address(_, City, ZipCode), _), City \= unknown.
% Question 3:
patients_in_delhi(Patient, Name) :-
        patient(Patient, Name, address(_, "Delhi", _), _).
% Question 4:

patients_treated_by_doctor(DocId, Name):-
        patient(Patient, Name, _, Treatments), member(treatment(DocId, _), Treatments).
```

```
% Question 5:
patients_with_disease(Disease, Patients) :-
findall(Patient, hypothesis(Patient, Disease), Patients).
% Question 6:
patient_addresses(Addresses) :-
        findall((BuildingName, CityCode), patient(_, _, address(BuildingName, _,
CityCode), _), Addresses).
% Question 7:
doctors_for_patient(Patient, Doctors) :-
        patient(Patient, _, _, Treatments),
        findall(Doctor, (member(treatment(DocId, _), Treatments), doctor(DocId, Doctor)),
Doctors).
```

Q2) Implement Traveling Salesman problem using Breadth First Search algorithms.

```
class TSP:
    def __init__(self):
        self.graph = []
        self.init_graph()
        self.bfs()

    def init_graph(self): #abcdefg
    # let's push weights
        self.graph.append([0, 12, 10, -1, -1, -1, 12]) # a
        self.graph.append([12, 0, 8, 12, -1, -1, -1]) # b
        self.graph.append([10, 8, 0, 11, 3, -1, 9]) #c
        self.graph.append([-1, 12, 11, 0, 11, 10, -1]) # d
        self.graph.append([-1, -1, 3, 11, 0, 6, 7]) #e
        self.graph.append([-1, -1, -1, 10, 6, 0, 9]) #f
        self.graph.append([12, -1, 9, -1, 7, 9, 0]) #g

    def bfs(self):
        res = []
        visited = [0, 0, 0, 0, 0, 0, 0]
        start = 0
        queue = []
        queue.append(start)
```

```
        while len(queue) != 0:
            top = queue[0]
            if visited[top] == 0:
                res.append(top)
                visited[top] = 1
                for idx, j in enumerate(self.graph[top]):
                    if j > 0 and visited[idx] == 0: queue.append(idx)
            queue.pop(0)
        cost = 0
        for i in range(len(self.graph) - 1):
            cost += self.graph[res[i]][res[i + 1]]
        res = [chr(i+97) for i in res]
        print("BFS : ", res,)
        print("Cost : ", cost)


if __name__ == '__main__':
  TSP()
```

```
jhajharia@Nehals-MacBook-Air Exam % python3 bfs.py
BFS :  ['a', 'b', 'c', 'g', 'd', 'e', 'f']
Cost :  45
jhajharia@Nehals-MacBook-Air Exam %
```