Assignment 1: SQL and Prolog

Part 1: SQL

In this assignment, you will work with a university database to retrieve and analyze data using SQL queries.

Submission Instructions:

Please submit the following items on Canvas:

1. A university.sql file containing your SQL queries.
2. A screenshot showing the output of your queries when executed in MySQL. You may put all the screenshots into a single PDF file.

Exercises

Exercise 0: Setting Up MySQL and Populating Data

Instructions:

1) Install MySQL if it is not already installed. Ensure the MySQL server is running.
2) Log in to MySQL: `mysql -u root -p`
3) Create and populate the database using the following commands:

```
CREATE DATABASE university_db;
USE university_db;

SOURCE <DDL.sql>;
# replace <DDL.sql> with the actual file path.
# If you are using MySQL Workbench, you can open the script file and execute it.

SOURCE <smallRelationsInsertFile.sql>
# replace .SQL file with the actual file path.
```

Both sql scripts can be found at Canvas.

Exercise 1: SQL Qeuries

1. (10 pts) Find the names of courses in the Computer science department which have 3 credits. Hint: Use the `course` table.
2. (10 pts) For the student with ID 76543, show the course id and the title of all courses registered for by the student. Hint: Use the `course` and `takes` tables.
3. (10 pts) Find the departments that have more than two instructors (not including two). Hint: Use the `instructor` table.

Part 2: Prolog

In this part, you will design a Prolog program that constructs valid garden layouts under a set of constraints. You will write predicates to generate candidate plans and to verify the constraints.

Submission Instructions:

Please submit the following items on Canvas:

1. The Prolog program file (`garden.pl`) containing all defined facts and rules.
2. A .txt file containing all the queries.
3. A screenshot showing the output of your queries when executed in Prolog. You may put all the screenshots into a single PDF file.

Problem Description:

You will construct a single-row (1D) flower garden with N plantings. Each planting contains one flower species. The garden is represented as a Prolog list.

Flower bed rules:

1. The garden consists of 1 row, each with N plantings (N **is at least 4**). One flower species occupies each planting. The Row is horizontal layed out with plantings from 1 at the left to N at the right. You will use a Prolog list to represent the row.

2) Flowers have Name, Size, Wet/Dry, Color.
3) A given flower species can only be used once per row.
4) No two adjacent plantings can have the same color flower.
5) No two adjacent plantings can have flowers whose size is more than one size difference. Sizes are small, med, tall so small next to small is fine, small next to medium is fine, but small next to tall is not.
6) The two outermost plantings (1 and N) are dry, the two innermost are wet, the ones in between (if there are any) can take either.
7) The properties of available flower species are defined below:.

```
flower(daisies, med, wet, yellow).
flower(roses, med, dry, red).
flower(petunias, med, wet, pink).
flower(daffodils, med, wet, yellow).
flower(begonias, tall, wet, white).
flower(snapdragons, tall, dry, red).
flower(marigolds, short, wet, yellow).
flower(gardenias, med, wet, red).
flower(gladiolas, tall, wet, red).
flower(bird_of_paradise, tall, wet, white).
flower(lilies, short, dry, white).
```

```
flower(azalea, med, dry, pink).
flower(buttercup, short, dry, yellow).
flower(poppy, med, dry, red).
flower(crocus, med, dry, orange).
flower(carnation, med, wet, white).
flower(tulip, short, wet, red).
flower(orchid, short, wet, white).
flower(chrysanthemum, tall, dry, pink).
flower(dahlia, med, wet, purple).
flower(geranium, short, dry, red).
flower(lavender, short, dry, purple).
flower(iris, tall, dry, purple).
flower(peonies, short, dry, pink).
flower(periwinkle, med, wet, purple).
flower(sunflower, tall, dry, yellow).
flower(violet, short, dry, purple).
flower(zinnia, short, wet, yellow).
```

Exercises

Implement the following predicates and provide at least two sample queries for testing each, along with the corresponding output.

1. (10 pts) plantassign(N, List) Construct a length-N list and assign a species to each position.
2. (10 pts) uniquecheck(List) Ensure no species is repeated.
3. (10 pts) colorcheck(List) Ensure adjacent plantings do not share the same color.
4. (10 pts) sizecheck(List) Ensure adjacent plantings differ by at most one size level.
5. (10 pts) wetcheck(N, List) Enforce the watering rule.
6. (10 pts) writegarden(List) Print the final garden plan in a readable format.
7. (10 pts) gardenplan(N, List) Orchestrate the assignment, check all constraints, and print the garden.

Notes:

1. In all predicates, the List argument refers to the same list of N elements, where each element is a flower species name (not a full flower(...) structure).
2. To implement each predicate, you may define appropriate supporting rules as needed.

Extra Exercises (15 bonus points)

Extend your solution to support a 2D flower garden with $N \times M$ plantings. Implement all of the functions listed above in the 2D setting:

1. plantassign(N, M, List) Construct a length-NxM list and assign a species to each position.
2. uniquecheck(List) Ensure no species is repeated.
3. colorcheck(List) Ensure adjacent plantings do not share the same color.
4. sizecheck(List) Ensure adjacent plantings differ by at most one size level.
5. wetcheck(N, M, List) Enforce the watering rule.
6. writegarden(List) Print the final garden plan in a readable format.
7. gardenplan(N, M, List) Orchestrate the assignment, check all constraints, and print the garden.