Introduction to Artificial Intelligence
Lecture 8: Exercises (With Solutions)

Problem:

A university department wants to represent and reason about students, their courses, and their academic performance using pyDatalog.

You are asked to build a logic-based knowledge system that supports inference, recursion, and numerical aggregation.

Given Information:

- The department offers the following courses:
  - AI (3 credits)
  - Math (4 credits)
- Students are enrolled and receive marks as follows:
  - Alice scored 85 in AI.
  - Alice scored 90 in Math.
  - John scored 60 in AI.
  - John scored 70 in Math.
  - Bob scored 40 in Math.
  - Bob scored 55 in AI.

Task 1:

1- Enter the above information as facts:

- Which student took which course.
- The mark obtained.
- the number of credits per course.

Declare all necessary variables and predicates properly.

Task 2:

1- Write pyDatalog queries to:

- Print all students with the courses they took and the corresponding marks.
- Find all students who took the course AI.
- Find the mark scored by Alice in Math.

Task 3:

1- Define the following rules:

- passed(Student, Course): A student passes a course if their mark is ≥ 50.
- failed(Student, Course): A student fails a course if their mark is < 50.

2- Use the rules to print all failed courses for each student.

Task 4:

1- Using aggregate functions, define:

- total_credits[Student]: The total number of credits for all courses a student has taken.
- course_count[Student]: The number of courses taken by each student.
- highest_mark[Student]: The highest mark obtained by each student.

2- Write queries to display these values for each student.

Task 5:

1- Define a logical function that computes the weighted sum of marks as:

weighted_score[Student] = sum over all courses of (mark × course_credits)

2- Query the function to compute the weighted score for Alice.

Task 6:

1- Define the following rule:

honor_student(Student): A student is considered an honor student if:

- Their rate is greater than 85%, **and** they have passed all courses they took.

2- Use a query to list all honor students.

Solution:

```
import pyDatalog.pyDatalog as py

# ------------------------------------------------
# Term declarations
# ------------------------------------------------
py.create_terms(
    'Student, Course, Mark, Credits, '
    'takes, score, course_credits, '
    'passed, failed, '
    'total_credits, course_count, highest_mark, '
    'weighted_score, honor_student, average_score,'
    'X, Y, Z, M, C, S,W,T'
)

# ------------------------------------------------
# Task 1:
# ------------------------------------------------
+course_credits('ai', 3)
+course_credits('math', 4)


+takes('Alice', 'ai')
+score('Alice', 'ai', 85)

+takes('Alice', 'math')
+score('Alice', 'math', 90)


+takes('John', 'ai')
+score('John', 'ai', 60)

+takes('John', 'math')
+score('John', 'math', 70)

+takes('Bob', 'math')
+score('Bob', 'math', 40)

+takes('Bob', 'ai')
+score('Bob', 'ai', 55)
```

```
# ------------------------------------------------
# Task 2:
# ------------------------------------------------

print("All scores:")
print(score(Student, Course, Mark))

print("All students who tool the ai course")
print(takes(X,'ai'))

print("mark scored by Alice in math")
print(score('Alice','ai',X))

# ------------------------------------------------
# Task 3:
# ------------------------------------------------

passed(Student, Course) <= score(Student, Course, Mark) & (Mark >= 50)
failed(Student, Course) <= score(Student, Course, Mark) & (Mark < 50)

print('all failed courses for each student')
print(failed(Student,Course))
```

```
# ------------------------------------------------
# Task 4:
# ------------------------------------------------

(total_credits[Student] == sum_(Credits,for_each=Course)) <= score(Student, Course, Mark) & course_credits(Course,Credits)

(course_count[Student] == len_(Course)) <= takes(Student, Course)

(highest_mark[Student] == max_(Mark, order_by=Mark)) <= score(Student, Course, Mark)

print('total credits for each student')
print(total_credits[Student]==X)

print('number of courses for each student')
print(course_count[Student]==X)

print('highest score for each student')
print(highest_mark[Student]==X)

# ------------------------------------------------
# Task 5:
# ------------------------------------------------

(weighted_score[Student] == sum_(Z,for_each=Course)) <= (
    score(Student, Course, Mark) &
    course_credits(Course, Credits) &
    (Z==Mark * Credits)
)


print('weighted score for Alice')
print(weighted_score['Alice']==X)
```

```
# ---------------------------------------------------
# Task 6:
# ---------------------------------------------------
(honor_student[Student] == True) <= (
    (weighted_score[Student] == S) &
     (total_credits[Student]==T) &
    (S/T > 85) &
    (~failed(Student, Course)))

print('Honor Students')
print(honor_student[Student]==X)
```