Written Assignment #7
Due Wednesday June 3

1. Consider the following ODL schema for a database of students applying for summer internships.

```
interface Student
(extent Students, key ID) {
   attribute integer ID;
   attribute Struct{string first, string last} name;
   relationship Set<Internship> applied
      inverse Internship::applicants;
  }

interface Internship
(extent Positions, key (company, city)) {
   attribute string company;
   attribute string city;
   relationship Set<Student> applicants
      inverse Student::applied;
  }
```

Write OQL queries for each of the following.

(a) Find the ID's of all students whose last name is Smith.

(b) Find the ID's and last names of all students who have applied to an internship at a company in Palo Alto. Do not repeat (ID,last-name) pairs in the result, even if the student has applied to many internships in Palo Alto.

(c) If you used `distinct` in your answer for part (b), rewrite the query so you don't need to use `distinct`. Conversely, if you didn't use `distinct` in your answer for part (b), rewrite the query so you do need to use `distinct` in order to guarantee that duplicates are eliminated.

(d) Find the names of all companies in Palo Alto such that at least one student S (say) with ID between 25 and 50 has applied for an internship at that company, and all internships student S has applied for are in Palo Alto or San Jose.

(e) Recall that the result of an OQL query or subquery is a set or a bag. OQL allows two sets (bags) to be compared using =, where two sets (bags) are equal if they contain exactly the same objects. Find all pairs of student ID's such that the two students have applied to internships at the exact same set of companies in Palo Alto. (The students may have applied to different internships at companies in other cities.) Return each pair of ID's exactly once, and order the final result based on the last name of the first student in each pair.

(f) Can you write the query in part (e) without using set or bag equality? If so, write it. If not, explain why not.

(over)

2. Consider the following self-explanatory relational schema that uses SQL3 row types and references.

```
create row type AddressType as (street string, city string)
create row type StudentType as (name string, address AddressType)
create row type CollegeType as (name string, city string)

create table Student of type StudentType
create table College of type CollegeType
create table Attends (student ref(StudentType), college ref(CollegeType),
                      tuition integer)
create table Roommates (student1 ref(StudentType), student2 ref(StudentType))
```

   Write SQL3 queries for each of the following. You may assume that student and college names are unique, that all students have exactly one address, attend one college, and have one roommate, and that all colleges are located in exactly one city.

   (a) Find the names of all students who live in Palo Alto.

   (b) Find the names of all students who attend Stanford.

   (c) Find the names of all students who live in the same city as the college they attend and pay tuition of at least $10,000.

   (d) Find the names of all students who live in the same city and on the same street as their roommate.

3. Remind us of the relational schema you're using for your PDA. Then modify your schema to use SQL3 row types and references wherever it makes sense. At the very least use at least one instance of a row type in order to create a structure-valued attribute, and at least two instances of references to row types.

4. (Optional – depends if we have time to cover recursion in SQL3 before the assignment is due.) Do parts (a)–(d) of Exercise 5.10.1 on pages 322–323 of the textbook.