Each multiple-choice quiz problem is based on a "root question," from which the system generates different correct and incorrect choices each time you take the quiz. Thus, you can test yourself on the same material multiple times. We strongly urge you to continue testing on each topic until you complete the quiz with a perfect score at least once. Simply click the "Reset" button at the bottom of the page for a new variant of the quiz.

After submitting your selections, the system will score your quiz, and for incorrect answers will provide an "explanation" (sometimes for correct ones too). These explanations should help you get the right answer the next time around. To prevent rapid-fire guessing, the system enforces a minimum of 10 minutes between each submission of solutions.

Multiple Choice

3/3 points (graded)

[Q1] Consider a table T(A) containing a set of positive integers with no duplicates, and the following recursive SQL query. Note that this query includes nonlinear recursion, which technically is not permitted in the strict SQL standard.

```
With Recursive Mystery(X,Y) As
   (Select A As X, A As Y From T
    Union
    Select m1.X, m2.Y
    From Mystery m1, Mystery m2
    Where m2.X = m1.Y + 1)
Select Max(Y-X) + 1 From Mystery
```

While the definition looks complicated, the query in fact computes a property of T that can be stated very succinctly. First try to determine what Mystery is computing from T. Then choose which of the following is a correct statement about the final query result.

- If T = { 2, 3, 4, 5, 10, 11, 12} then the query returns 11.
- If T = {1, 3, 4, 5, 10, 11, 12} then the query returns 2.

- If T = {1, 5, 9, 10, 12, 15} then the query returns 15.
- If T = {7, 9, 10, 14, 15, 16, 18} then the query returns 3.

[Q2] Consider a relation Manager(manager,employee) where a tuple (m,e) in Manager specifies that person m is the manager of person e. The only key for Manager is both attributes together. The following recursive SQL query computes the relation Peer(X,Y).

```
With Recursive Peer(X,Y) As
   (Select M1.employee, M2.employee
    From Manager M1, Manager M2
    Where M1.manager = M2.manager AND M1.employee < M2.employee
    Union
    Select M1.employee, M2.employee
    From Peer S, Manager M1, Manager M2
    Where S.X = M1.manager AND S.Y = M2.manager
    And M1.employee < M2.employee)
Select * from Peer
```

Suppose the tuples in Manager are: (10, 9), (10,8), (9,7), (9,6), (8,6), (8,5), (7,4), (7,3), (6,3), (6,2), (5,2), (5,1). Consider the computation of Peer in the recursive query. Let the base case -- the first term of the Union -- be "round 1." Let each subsequent round of the recursion be "round 2," "round 3," and so on. Which of the following is a correct statement about when a pair gets added to Peer? (You may find it helpful to draw a figure that shows the Manager relationships.)

- (1,3) is added in round 1.
- (5,7) is added in round 1.
- (5,7) is added in round 2.

- (1,4) is added in round 2.

[Q3] Consider a relation Parent(par,child), where a tuple (p,c) in Parent specifies that person p is the parent of person c. The only key for Parent is both attributes together. We are interested in writing a recursive SQL query to find all descendants of the person named "Eve." Here are six possible definitions of a recursive relation Ancestor(X,Y). Note that some of the definitions include nonlinear recursion, which technically is not permitted in the strict SQL standard.

```
1: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent
       Union
       Select Ancestor.X, Parent.child
       From Ancestor, Parent
       Where Ancestor.Y = Parent.par)
2: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent Where par = 'Eve'
       Union
       Select Ancestor.X, Parent.child
       From Ancestor, Parent
       Where Ancestor.Y = Parent.par)
3: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent
       Union
       Select Parent.par, Ancestor.Y
       From Parent, Ancestor
       Where Parent.child = Ancestor.X)
4: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent
       Union
       Select Parent.par, Ancestor.Y
       From Parent, Ancestor
       Where Parent.child = Ancestor.X and Parent.par = 'Eve')
5: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent Where par = 'Eve'
       Union
       Select A1.X, A2.Y
       From Ancestor A1, Ancestor A2
       Where A1.Y = A2.X)
6: With Recursive Ancestor(X,Y) As
      (Select par, child From Parent
       Union
       Select A1.X, A2.Y
       From Ancestor A1, Ancestor A2
       Where A1.Y = A2.X and A1.X = 'Eve')
```

Consider two possible queries that can be used to complete any of the WITH statements (1)-(6):

```
A: Select Y From Ancestor
B: Select Y From Ancestor Where X = 'Eve'
```

Which of the following combinations correctly computes the descendants of "Eve"?

- (A) gives the correct answer with (2) and (3) only.
- Neither (A) nor (B) give the correct answer with (4) and (5).
- (B) gives the correct answer with (1), (2), and (3) only.
- Neither (A) nor (B) give the correct answer with (6).

Submit