CS 61A Exam Prep Session 1

SQL and Exam-prep strategy review

Jacob Wu // 11.28.2021 // Intended for Zody

SQL

SQL
Basics

- SELECT
- DISTINCT
- FROM
- UNION
- WHERE
- IN
- ORDER BY

SQL

Example Structure

- SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order]

- WITH ages AS (SELECT age FROM Penguin WHERE age > 10 ORDER BY name)
- SELECT * FROM Penguin WHERE age IN ages;

SQL

Exam Questions

6. (8 points) Six Degrees of Separation

You've probably heard that we are all within "six degrees of separation." That is, either we are friends (one degree), friends of friends (two degrees), friends of friends of friends (three degrees), etc. up to six degrees. We may, of course, be separated by several different distances, as when our friend is also a friend of a friend. Although there are obviously many different paths leading from you back to yourself, however, we won't consider you as being connected with yourself.

Suppose that `friends` is an SQL table with two columns, `F1` and `F2`, where in each row, `F1` and `F2` are the names of two friends—i.e., two people with one degree of separation between them. To make life easier, we'll assume that if (Peter, Paul) is in the table, then so is (Paul, Peter). We would like an SQL query that produces a two-column table named `linked` of people separated (by some chain of friends) by N or fewer degrees of separation, where N is some integer. In your solution, use '`N`' as if it is an integer literal, like 6. (The idea of using '`N`' instead of a specific number is to force your solution to be general.) Each pair in the resulting table should appear exactly once, with the name in the first column being first in alphabetical order.

For example, suppose that $N = 2$, then given the `friends` table on the left, we should get the `linked` table on the right, in some order. (The column names don't matter for `linked`, and so are not shown.)

friends

F1	**F2**
Peter	Paul
Jack	Paul
Rose	Jack
Paul	Sam
Cindy	Rose
Paul	Peter
Paul	Jack
Jack	Rose
Sam	Paul
Rose	Cindy

linked

Cindy	Rose
Cindy	Jack
Jack	Paul
Jack	Rose
Jack	Peter
Jack	Sam
Paul	Peter
Paul	Sam
Paul	Rose
Peter	Sam

SQL
Exam Questions

```
create table linked as
    with sep(S1, S2, degrees) as (
       select ________________________________________________ union
       select ___________________________________ from friends, sep
           where ____________________________________________________________
    )
select ________________ from ___________ where ______________________________;
```

SQL
Exam Questions

```
create table linked as
    with sep(S1, S2, degrees) as (
        select ________________________________________ union
        select ____________________________ from friends, sep
            where ________________________________________________
    )
select distinct S1, S2 from sep where S1 < S2;
```

① Start with what we have

SQL

Exam Questions

②Consider what we want

```
create table linked as

    with sep(S1, S2, degrees) as (

        select F1, F2, 1 from friends union

        select F1, S2, degrees +1 from friends, sep

            where ____________________________________
    )

select distinct S1, S2 from sep where S1 < S2;
```

①Start with what we have

SQL

Exam Questions

```
create table linked as
    with sep(S1, S2, degrees) as (
        select F1, F2, 1 from friends union
        select F1, S2, degrees +1 from friends, sep
            where degrees <= N and F2 = S1
    )
select distinct S1, S2 from sep where S1 < S2;
```

② Consider what we want

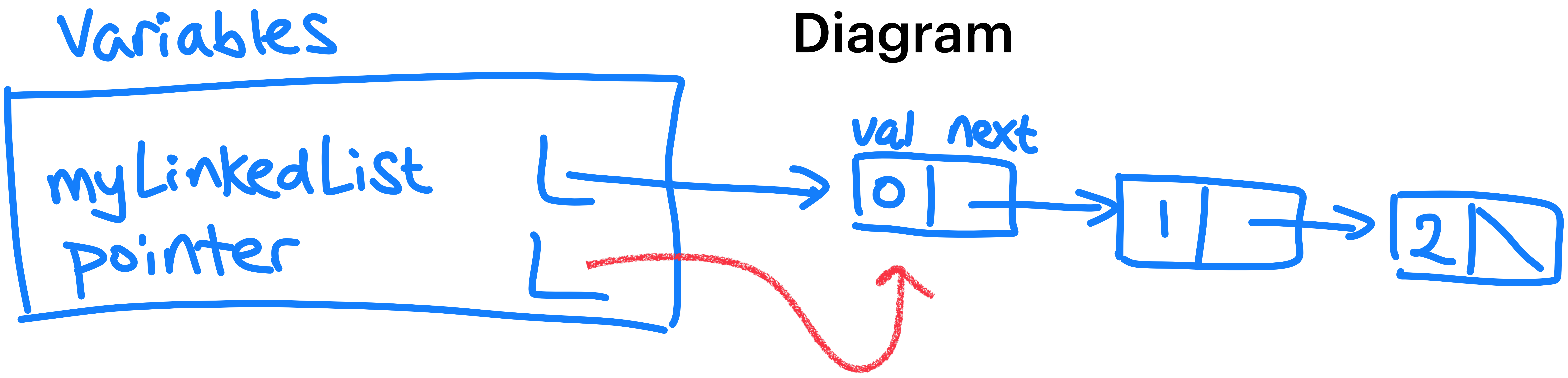
③ Set good conditions

① Start with what we have

Exam Prep

Linked List

Diagram

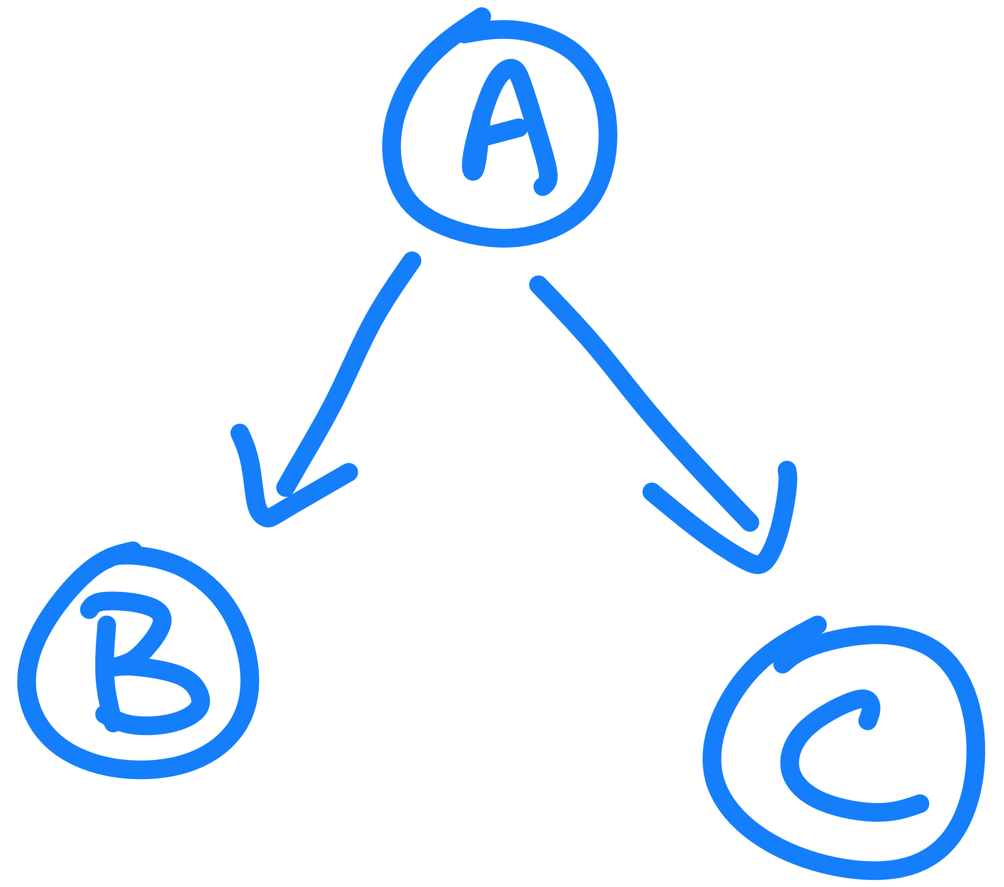


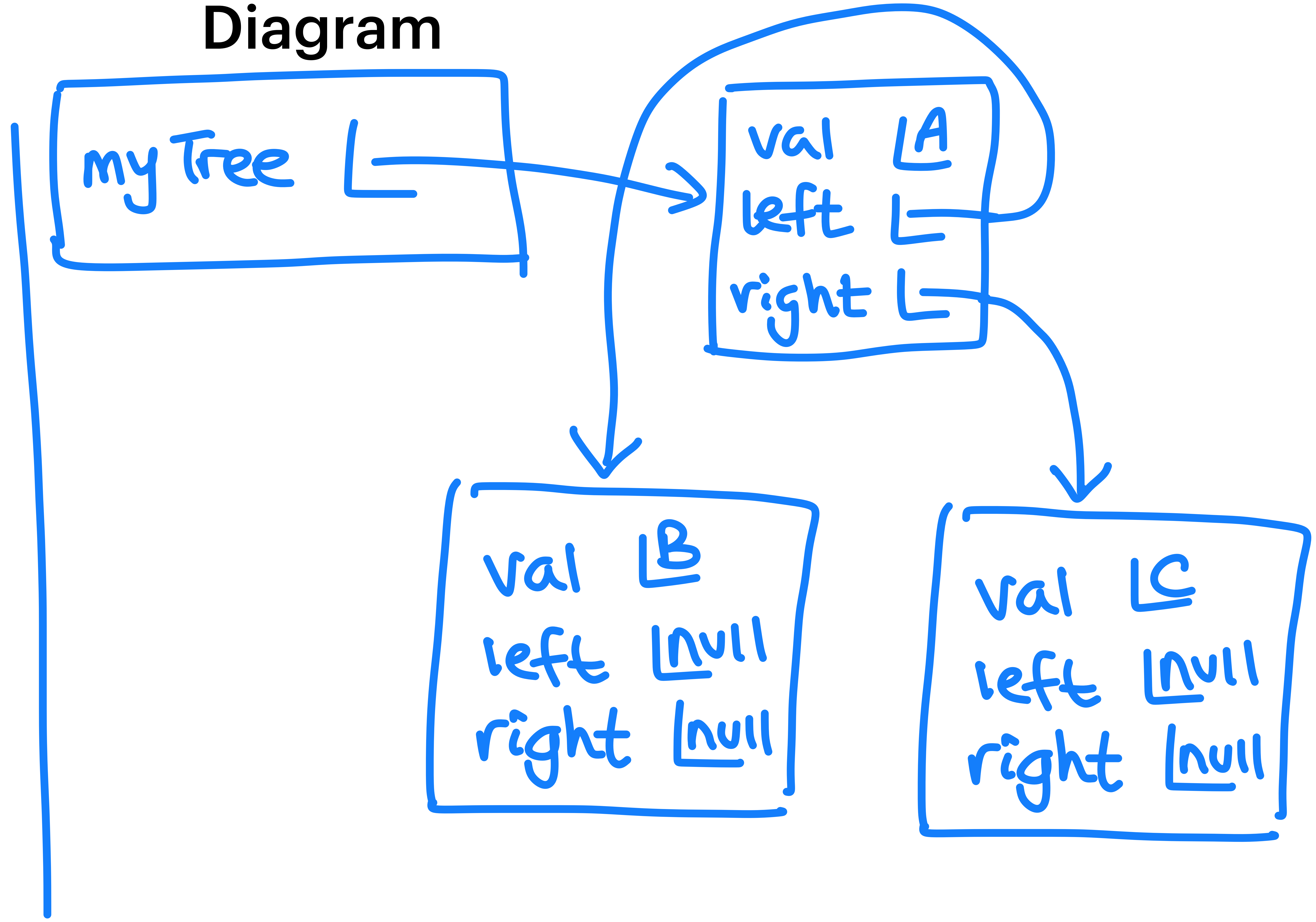
Linked List

- while current.rest is not Link.empty:
    - last.rest = Link(current.first + current.rest.first)
    - last, current = last.rest, current.rest
- last.rest = Link(1)

Tree

Diagram





Tree

- def print_column(tree, col):
  - def print_inner(t, k):
    - if t is BinTree.empty: return
    - if k == col: print(t.label)
    - else:
      - print_inner(t.left, k-1)
      - print_inner(t.right, k+1)
  - print_inner(tree, 0)

Iterator & Generator

Diagram

- Iterable (objects) can be used in a for loop, e.g. range(4), [1,2,3], {‘a’: 1}
- Generators simplify the creation of an iterator, e.g. functions with yield

```python
class yrange:
    def __init__(self, n):
        self.i = 0
        self.n = n

    def __iter__(self):
        return self

    def __next__(self):
        if self.i < self.n:
            i = self.i
            self.i += 1
            return i
        else:
            raise StopIteration()
```

```python
def yrange(n):
    i = 0
    while i < n:
        yield i
        i += 1
```

Iterator & Generator

- def amplify(f, x):
    - while x:
        - yield x
    - x = f(x)

Lambda

Review

- Lambda as small, anonymous function
- x = lambda a : a + 10
- x = lambda a, b : a * b
- x = lambda a, b, c : a + b + c

Lambda

- def multigroup(f, s):
  - def using(g, s):
    - if len(s) == 1:   return s[0]
    - else:
      - grouped = group(g, s)
      - return using(lambda x: f(g(x[0])), grouped)
    - return using(lambda x: x, s)