Iterators & Generators

1. Lazy Sunday (Fa14 Final Q4a)

(4 pt) A flat-map operation maps a function over a sequence and flattens the result. Implement the flat_map method of the FlatMapper class. You may use at most 3 lines of code, indented however you choose.

class FlatMapper:
    """A FlatMapper takes a function fn that returns an iterable value. The flat_map method takes an iterable s and returns a generator over all values that are within the iterables returned by calling fn on each element of s."

    >>> stutter = lambda x: [x, x]
    >>> m = FlatMapper(stutter)
    >>> g = m.flat_map((2, 3, 4, 5))
    >>> type(g)
    <class 'generator'>
    >>> list(g)
    [2, 2, 3, 3, 4, 4, 5, 5]
    """

def __init__(self, fn):
    self.fn = fn

def flat_map(self, s):
    for x in s:
        for r in self.fn(x):
            yield r
2. From the Other Side (Fa15 Final Q1)

```python
class Adele:
    times = '1000'
    def __init__(self, you):
        self.call = you
    def __str__(self):
        return self.times

class Hello(Adele):
    def __next__(self):
        return next(self.call)

never = iter('scheme2Bhome')

def any(more):
    next(never)
    print(outside)
    yield next(never)
    print(next(never))
    yield more(more)

outside = Hello(any(any))
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>'a'</td>
</tr>
<tr>
<td>iter('a')</td>
<td>Iterator</td>
</tr>
<tr>
<td>print('a') + 1</td>
<td>a</td>
</tr>
<tr>
<td>next(never)</td>
<td>'s'</td>
</tr>
<tr>
<td>next(outside)</td>
<td>1000 'h'</td>
</tr>
<tr>
<td>next(next(outside))</td>
<td>e 1000 'g'</td>
</tr>
<tr>
<td>list(never)[:3]</td>
<td>['2', 'B', 'h']</td>
</tr>
<tr>
<td>next(next(outside))</td>
<td>Exception</td>
</tr>
</tbody>
</table>

3. Apply That Again (Sp15 Final Q4a)

(4 pt) Implement amplify, a generator function that takes a one-argument function \( f \) and a starting value \( x \). The element at index \( k \) that it yields (starting at 0) is the result of applying \( f \) \( k \) times to \( x \). It terminates whenever the next value it would yield is a false value, such as 0, '', [], False etc.

```python
def amplify(f, x):
    """Yield the longest sequence \( x, f(x), f(f(x)), ... \) that are all true values."

>>> list(amplify(lambda s: s[1:], 'boxes'))
['boxes', 'oxes', 'xes', 'es', 's']

>>> list(amplify(lambda x: x//2-1, 14))
[14, 6, 2]

while x:
    yield x

x = f(x)
```
SQL

4. Highly Exclusive (Fa15 Final Q7c)

(4 pt) Select all positive integers that have at least 3 proper multiples that are less than or equal to X. A proper multiple m of n is an integer larger than n such that n evenly divides m (≡ m % n == 0).

The resulting table should have two columns. Each row contains an integer (that has at least 3 proper multiples) and the number of its proper multiples up to X. For example, the integer 3 has 5 proper multiples up to 20: 6, 9, 12, 15, and 18. Therefore, 315 is a row. There are five rows in the table when X is 20: 1, 19, 219, 315, 414, and 513. Your statement must work correctly even if X changes to another constant (such as 30) to receive full credit.

```
create table X as select 20 as X;
with ints(n) as (select 1 union select n+1 from ints, X where n < X)

select b.n, count(*) from ints as a, ints as b
where a.n > b.n and a.n % b.n = 0
  group by b.n having count(*) > 2;
```

5. Counting Stars (Su15 Final 7b)

(2 pt) When the Berts eat at a restaurant, they record a review in a SQL table called reviews:

<table>
<thead>
<tr>
<th>restaurant</th>
<th>user</th>
<th>stars</th>
<th>review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barney’s</td>
<td>Albert</td>
<td>4</td>
<td>Used to like it</td>
</tr>
<tr>
<td>Chipotle</td>
<td>Robert</td>
<td>5</td>
<td>BOGO! BOGO!</td>
</tr>
<tr>
<td>Eureka</td>
<td>Albert</td>
<td>5</td>
<td>My favorite</td>
</tr>
<tr>
<td>Bongo Burger</td>
<td>Albert</td>
<td>2</td>
<td>When I’m desperate</td>
</tr>
<tr>
<td>Umami Burger</td>
<td>Albert</td>
<td>5</td>
<td>I love truffle fries!</td>
</tr>
</tbody>
</table>

Write an SQL query to figure out how many restaurants Albert gave 4 or 5 stars. Using the table above, the output to your query should be the following:

<table>
<thead>
<tr>
<th>stars</th>
<th>number of reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

```
select stars, count(*) from reviews
where user = "Albert"
  group by stars
having stars >= 4;
```
6. Anagrams (Fa17 Quiz 11)
Create a table anagrams that contains all the anagrams of a word like cats. An anagram is a rearrangement of the letters in a word. For example, tacs and sact are anagrams of cats.

*Hint:* Each letter must be used exactly once, so the sum of the positions should equal 1111.

```sql
CREATE TABLE anagrams as

WITH word(letter, position) AS (  
    SELECT 'c', 1 UNION  
    SELECT 'a', 10 UNION  
    SELECT 't', 100 UNION  
    SELECT 's', 1000  
)

SELECT a.letter || b.letter || c.letter || d.letter

FROM word AS a, word AS b, word AS c, word AS d

WHERE a.position + b.position + c.position + d.position = 1111;

SELECT * FROM anagrams;
tacs
sact
...
ctsa
atsc