[CS 11] Prac 11f – One-To-One Correspondence

Problem Statement

The agents and spies of Westalian Intelligence Services' Eastern-Focused Division (WISE) need to communicate with each other in secret.

WISE has **agents** all over Westalia and **spies** all over Ostania. Each spy in Ostania corresponds with a unique agent in Westalia, and each agent in Westalia corresponds with a unique spy in Ostania. They correspond with each other on a regular basis via letters delivered covertly.

Each spy and agent has a code name. Spies have different code names with each other. Agents have different code names with each other. However, there could be a spy and an agent with the same name.

From time to time, new agent-spy pairs are deployed, and some old agent-spy pairs are retired.

Your task is to maintain the status of all the agent-spy pairs across time, given a number of these events.

Task Details

Your task is to implement a *class* called `OneToOneCorrespondence`. An instance of this class must maintain spy-agent pairs across multiple operations.

Its constructor must take in a variable number of pairs of `str`s denoting the initial agent-spy pairs:

- the first element of the pair is the agent's name.
- the second element of the pair is the spy's name.

The constructor must raise a `ValueError` if a name appears multiple times as an agent's name in different pairs, or if a name appears multiple times as a spy's name in different pairs.

The class must implement the following methods:

- `.get_right(v)` where `v` is a `str` denoting an agent's name. This must return a `str` denoting the corresponding spy's name of the pair whose agent has name `v`. If there is no such pair, this method must raise a `ValueError`.
- `.get_left(v)` where `v` is a `str` denoting a spy's name. This must return a `str` denoting the corresponding agent's name of the pair whose spy has name `v`. If there is no such pair, this method must raise a `ValueError`.
- `.set_pair(l, r)` where `l` and `r` are `str`s denoting an agent's name and a spy's name, respectively. This must add (deploy) the said agent-spy pair to the data. If there's already a pair with agent `l` or spy `r`, this method must raise a `ValueError`. This function must return `None`.
- `.remove_pair_with_left(v)` where `v` is a `str` denoting an agent's name. This must remove (retire) the pair whose agent has name `v`. If there no such pair, this method must raise a `ValueError`. This function must return `None`.
- `.remove_pair_with_right(v)` where `v` is a `str` denoting a spy's name. This must remove (retire) the pair whose spy has name `v`. If there no such pair, this method must raise a `ValueError`. This function must return `None`.

Restrictions

(See 11a for more restrictions)

For this problem in particular:

- The source code limit is 3000.

Example Testing

Note: This assumes that your submission has filename `prac11f.py`. Write this testing code in a separate file, say `test_prac11f.py`, and run it to test your code.

Example 1 Testing

Copy

```
from contextlib import contextmanager

from prac11f import OneToOneCorrespondence

@contextmanager
def raises_value_error():
    """ context block must raise a ValueError """
    try:
        yield
    except ValueError:
        pass
    else:
        assert False


data = OneToOneCorrespondence(
    ('K', 'J'),
    ('J', 'T'),
    ('O', 'Z'),
    ('L', 'AA'),
    ('M', 'Frank'),
)

assert data.get_left('J') == 'K'
assert data.get_right('J') == 'T'
assert data.get_right('L') == 'AA'
data.set_pair('AA', 'A')
assert data.get_right('AA') == 'A'
data.remove_pair_with_right('Frank')
with raises_value_error():
    data.get_left('Frank')
data.set_pair('Frank', 'M')
with raises_value_error():
    data.set_pair('Frank', 'HighT')
```

Constraints

Let n be the total number of pairs passed to the constructor. Let q be the total number of calls made to the methods.

- The class `OneToOneCorrespondence` will be instantiated at most $70{,}000$ times.
- $0 \leq n \leq 350{,}000$
- The sum of ns is at most $350{,}000$
- The total number of method calls is at most $350{,}000$.
- Each name is a string of at most 5 (uppercase or lowercase) letters.

Scoring

- You get 100 ❤️ points if you solve all test cases where:
  - $n \leq 50$
  - $q \leq 600$
  - The sum of ns is at most 600.
  - no function call will be made that needs to raise an exception.
- You get 40 ❤️ points if you solve all test cases where:
  - $n \leq 50$
  - $q \leq 600$
  - The sum of ns is at most 600.
- You get 30 ❤️ points if you solve all test cases where:
  - $n \leq 5{,}000$
  - $q \leq 5{,}000$
  - The sum of ns is at most $5{,}000$.
- You get 30 ❤️ points if you solve all test cases.

Submit solution

[CS 11] Practice 11

Points: 200 (partial)

Time limit: 8.0s

Memory limit: 1G

Author: kvatienza (Kevin Atienza)

Problem type

Allowed languages ~~NONE~~, py3

Clarifications

Report an issue

No clarifications have been made at this time.