Coop Ordering

A large group of families orders groceries together, known as a cooperative or *coop*. They have asked us to build a simple ordering system.

The coop allows families to choose from a collection of recipes. Each recipe uses a subset of the ingredients offered by the coop. The amounts for each ingredient will be expressed as a whole number. (We won't worry about the units). For example, here is the ingredient list for a recipe for making a simple *salad*:

Ingredient	Amount
lettuce	1
pepper	1
carrot	2
cherry-tomato	12

And here is the ingredient list for a recipe for making roasted carrots:

Ingredient	Amount
carrot	12
honey	1
butter	1

When it is time for the coop to put together an order, each family will report the ingredients they already have in their pantry. For example, the Smith family might have the following ingredients in their pantry:

Ingredient	Amount
ketchup	1
carrot	1
pepper	2
butter	1
ground-beef	5

Please note that a family may have zero ingredients in their pantry.

Each family will also specify the recipes they wish to make.

Your task is to determine the ingredients the coop needs to order this week to allow each family to make the recipes they have chosen. The coop does not keep any ingredients on hand and does not order any extra ingredients. Also, a family can only use ingredients from their own pantry; they cannot use ingredients from another family's pantry.

For example, if coop consisted solely of the Smith family and they wanted to make the salad and roasted carrots shown above, then the coop would need to order:

Ingredient	Amount
carrot	13
cherry-tomatoes	12
honey	1
lettuce	1

Notice that the list of items the coop needs to order does not include peppers or butter even though the salad recipe requires one pepper and the roasted carrots requires one butter. Why? Because the Smith's have a pepper and a butter on hand. Also, notice that it includes 13 carrots, because the salad recipe calls for two and the roasted carrot recipe requires 12 (for a total of 14 carrots needed) and the Smith's only have one on hand.

Provided Classes

We have defined two classes for use in this task: `Ingredient`, which will be used to represent ingredients, and `Order`, which will be used to represent orders. Instance of the `Ingredient` class will have three public attributes:

- `name`: the name of the ingredient (e.g., carrot, butter, etc),
- `key`: the name of a family, if the instance is representing an ingredient from a family's pantry, or the name of a recipe, if the ingredient is representing an ingredient in a recipe, and
- `amount`: the amount of the ingredient.

Instances of the `Order` class will have two public attributes:

- `family_name`: the name of a family and
- `recipe_name`: the name of the recipe the family wishes to make.

Task

Your task is to complete the function `mk_coop_order`. The function will take a list of pantry ingredients, a list of recipe ingredients, and a list of orders. You should not make any assumptions about the order ingredients appear in the lists. That is, you are **not** guaranteed that all the ingredients for a single family (or recipe) occur together. The function should return a map (i.e. dictionary, hash table, hashmap, etc) that maps ingredient names (represented as strings) to amounts (represented as integers) that contains information about what the coop needs to purchase for the week.

Requirements

Your solution must be efficient under the following assumptions:

1. a large number of families participate in the coop,
2. the number of recipes supported by the coop is large, and
3. the number of ingredients used across all the recipes is large.

When processing a single order, your solution should take time proportional to the number of ingredients in the recipe.

You must solve the problem in **one** of Python 3, Java, or C++. See below for language-specific instructions and starter code.

For the practice problems, we have provided the necessary files for you. When you take the actual exam, you will be expected to copy code into files with specific names.

Python3

Your task is to complete the function `mk_coop_order` in `coop.py`

Here is the code for the `Ingredient` class:

```
'''
Ingredient class module for the coop problem.
'''

class Ingredient:
    '''
    Ingredient: represent an ingredient in a family's pantry, an
    ingredient needed for a recipe, or an ingredient in the
    coop's ledger.
    '''
    def __init__(self, name: str, key: str, amount: int) -> None:
        '''
        Initialize an Ingredient object.

        Arguments:
            name: The name of the ingredient.
            key: The family or recipe key for the ingredient.
            amount: The amount of the ingredient.

        Returns:
            None
        '''
        self.name = name
        self.key = key
        self.amount = amount

    def to_string(self) -> str:
        '''
        Returns a string representation of the Ingredient object.

        Arguments:
            None

        Returns:
            A string representation of the Ingredient object.
        '''
        return f'item: {self._name}, key: {self._key}, amount: {self._amount}'
```

You can find this code in a file named `ingredient.py`.

Here is the code for the `Order` class:

```
'''
Order class module for the coop problem.
'''

class Order:
    '''
    Represents a family's order, which is composed of a family name and a recipe
    name.
    '''
    def __init__(self, family: str, recipe: str) -> None:
        '''
        Initialize an Order object.

        Arguments:
            family: The family name.
            recipe: The recipe name.

        Returns:
            None
        '''
        self.family = family
        self.recipe = recipe

    def to_string(self) -> str:
        '''
        Returns a string representation of the Order object.

        Arguments:
            None

        Returns:
            A string representation of the Order object.
        '''
        return f'order: {self._family} {self._recipe}'
```

You can find this code in a file named `order.py`.

And here is the skeleton code this task:

```
'''
Co-op module for the coop problem.
'''

from typing import List, Dict
from ingredient import Ingredient
from order import Order


def mk_coop_order(pantry_items: List[Ingredient],
                  recipe_items: List[Ingredient],
                  orders: List[Order]) -> Dict[str, int]:
    '''
    Takes in a list of pantry items, a list of recipe items, and a list of
    orders and returns a dictionary of the items to be ordered from the
    Co-op and the quantity of each item to be ordered.

    Arguments:
        pantry_items: A list of Ingredient objects representing the items
            currently in the pantry.
        recipe_items: A list of Ingredient objects representing the items
            needed to make the recipe.
        orders: A list of Order objects representing the orders that have
            been placed.

    Returns:
        A dictionary of the items to be ordered from the Co-op and the
        quantity of each item to be ordered.
    '''
    # TODO: Implement this function
    return None
```

You can find this code in a file named `coop.py`.

Your implementation will need to use the `Ingredient` and `Order` classes.
You must only modify and submit file `coop.py`

Java

Your task is to complete the function `mkCoopOrder` in `Coop.java`

Here is the code for the `Ingredient` class:

```
/**
 * Ingredient: represent an ingrediant in a family's pantry, an
 *   ingredient needed for a recipe, or an ingredient in the
 *   coop's ledger.
 **/
public class Ingredient {
    String name;        // the name of the ingredient
    String key;         // a family name or recipe name
    int amount;

    public Ingredient(String ingredient_name,
                      String key,
                      int amount) {
        this.name = ingredient_name;
        this.key = key;
        this.amount = amount;
    }

    public String toString() {
        return "item: " + this.key + " " + this.name + " " + this.amount;
    }
}
```

You can find this code in a file named `Ingredient.java`.

Here is the skeleton code for the `Order` class:

```
/**
 * Order: represents a order for the coop problem.
 **/

public class Order {
    public String family_name;
    public String recipe_name;

    public Order(String family, String recipe) {
        this.family_name = family;
        this.recipe_name = recipe;
    }


    public String toString() {
        return "order: " + this.family_name + " " + this.recipe_name;
    }
}
```

You can find this code in a file named `Order.java`.

And here is the skeleton code this task:

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.HashMap;

public class Coop {

    public static Map<String, Integer> mkCoopOrder(List<Ingredient> pantry_items,
                                                   List<Ingredient > recipe_items,
                                                   List<Order> orders) {
        // TODO: complete this function
        // return included to allow the skeleton code to compile
        return null;
    }
}
```

You can find this code in a file named `Coop.java`

Your implementation will need to use the `Ingredient` and `Order` classes.
You must only modify and submit file `Coop.java`

C++

Your task is to complete the function `mk_coop_order` in `Coop.cpp`.

Here is the header file for the `Ingredient` class:

```cpp
#ifndef _Ingredient_
#define _Ingredient_
#include <string>

/**
 * Ingredient: represent an ingrediant in a family's pantry, an
 *   ingredient needed for a recipe, or an ingredient in the
 *   coop's ledger.
 **/

class Ingredient {
public:

std::string name; // the name of the ingredient
std::string key; // a family name or recipe name
int amount;

Ingredient (
    std::string const &ingredient_name,
    std::string const &_key,
    int const &_amount)
: name(ingredient_name), key(_key), amount(_amount) {}

std::string to_string() {
    return "item: " + key + " " + name + " " + std::to_string(amount);
}

};

#endif
```

You can find this code in a file named `Ingredient.h`. This header file contains all the code necessary for the `Ingredient` class; there is no need for a separate implementation file.

Here is the header file for the `Order` class:

```
#ifndef _Order_
#define _Order_
#include <string>

class Order {
public:

std::string family_name;
std::string recipe_name;

Order (std::string const &family, std::string const &recipe)
: family_name(family), recipe_name(recipe) {}

std::string to_string() {
    return "order: " + family_name + " " + recipe_name;
}

};

#endif
```

You can find this code in a file named `Order.h`. As with `Ingredient`, the header file contains all the code necessary for the `Order` class; there is no need for a separate implementation file.

Finally, here is the header code this task:

```
#ifndef _Coop_
#define _Coop_

#include "Ingredient.h"
```

```
#include "Order.h"
#include <string>
#include <vector>
#include <unordered_map>

std::unordered_map<std::string, int> mk_coop_order(
    std::vector<Ingredient*> const &pantry_items,
    std::vector<Ingredient*> const &recipe_items,
    std::vector<Order*> const &orders
);

#endif
```

You can find this code in a file named `Coop.h`.

Here is the skeleton code this task:

```
#include "Coop.h"
#include "Ingredient.h"
#include "Order.h"
#include <string>
#include <vector>
#include <unordered_map>
#include <utility>
#include <iostream>

std::unordered_map<std::string, int> mk_coop_order (
    std::vector<Ingredient*> const &pantry_items,
    std::vector<Ingredient*> const &recipe_items,
    std::vector<Order*> const &orders
) {
    // TODO
    return std::unordered_map<std::string, int>();
}
```

You can find this code in a file named `Coop.cpp`.

You must only modify and submit file `Coop.cpp` Do not include a `main` function in the `Coop.cpp` file, as that will interfere with our automated tests. If you want to test your implementation informally, please do so by creating a separate file containing a `main` function.