Class Exercise:

Collections

You are tasked with developing an advanced task management system that can handle multiple users, prioritize tasks, and track task completion. The system will be used by a project management team to coordinate tasks and monitor progress.

Your task is to create a class called TaskManager that will utilize the following data structures and classes:

* ChainMap: This data structure will be used to combine multiple dictionaries into a single dictionary. In our case, we will use it to combine user-wise task dictionaries into a single dictionary.
* Counter: This class will help us count the number of tasks in each category for each user. It will be used to keep track of the task counts for each user and category.
* deque: This data structure will be used to maintain a queue of tasks. Whenever a new task is created, it will be added to the deque. When a task is completed or removed, it will be removed from the deque.
* defaultdict: This class will be used to create a dictionary with a default value. In our case, we will use it to create a dictionary that automatically assigns an empty deque to a new user when it is encountered for the first time.
* namedtuple(): This function will be used to create a class called Task that will represent a single task. It will have attributes such as name, user, category, description, and priority.
* OrderedDict: This class will be used to create an ordered dictionary that will preserve the order of task creation. Whenever a new task is added, it will be appended to the end of the ordered dictionary.

Now, create the TaskManager class with the following methods:

* **\_\_init\_\_():** Initializes the task manager by creating the necessary data structures.
* add\_task(task): Adds a new task to the task manager. The task should be an instance of the Task class. The method should update the user-wise task dictionary, increment the task count for the corresponding user and category, and add the task to the end of the ordered dictionary.
* remove\_task(task\_name): Removes a task from the task manager based on its name. The method should remove the task from the user-wise task dictionary, decrement the task count for the corresponding user and category, and remove the task from the ordered dictionary.
* get\_task\_count(user, category): Returns the number of tasks in a given category for a specific user.
* get\_task\_list(user, category): Returns a list of tasks in a given category for a specific user, ordered by their creation.
* complete\_task(task\_name): Marks a task as completed. The method should remove the task from the user-wise task dictionary, decrement the task count for the corresponding user and category, and remove the task from the ordered dictionary.
* get\_completed\_tasks(user): Returns a list of completed tasks for a specific user, ordered by their completion.

Note: Remember to import the necessary modules and classes at the beginning of your code.

from collections import ChainMap, Counter, deque, defaultdict, namedtuple, OrderedDict

Task = namedtuple('Task', ['name', 'user', 'category', 'description', 'priority'])

class TaskManager:

    def \_\_init\_\_(self):

# Your code here

    def add\_task(self, task):

# Your code here

    def remove\_task(self, task\_name):

# Your code here

    def get\_task\_count(self, user, category):

# Your code here

    def get\_task\_list(self, user, category):

# Your code here

    def complete\_task(self, task\_name):

# Your code here

    def get\_completed\_tasks(self, user):

# Your code here

# Testing the TaskManager class

tm = TaskManager()

# Adding tasks

task1 = Task('Task 1', 'User A', 'Category A', 'Do something', 'High')

task2 = Task('Task 2', 'User B', 'Category B', 'Do something else', 'Low')

task3 = Task('Task 3', 'User A', 'Category A', 'Do another thing', 'Medium')

tm.add\_task(task1)

tm.add\_task(task2)

tm.add\_task(task3)

# Getting task counts

# Your code here

# Getting task lists

# Your code here

# Completing a task

tm.complete\_task('Task 1')

# Getting updated task counts and lists

# Your code here

# Getting completed tasks

Expected output:

2

1

[Task(name='Task 1', user='User A', category='Category A', description='Do something', priority='High'), Task(name='Task 3', user='User A', category='Category A', description='Do another thing', priority='Medium')]

[Task(name='Task 2', user='User B', category='Category B', description='Do something else', priority='Low')]

1

[Task(name='Task 3', user='User A', category='Category A', description='Do another thing', priority='Medium')]

[Task(name='Task 1', user='User A', category='Category A', description='Do something', priority='High')]