Instructions – Midterm Practice – Recipe Manager

Overview:
This assignment is designed to test OOP concepts, focusing on inheritance and the use of multiple classes. You will be creating a program that simulates cooking scenarios where cooks can learn different recipes and display their culinary repertoire. To explore OOP inheritance, you will also create a subclass for expert cooks who can prepare meals in half the preparation time. You also need to understand how to access private variables.

Libraries Required:
- random

Classes Required:
- *Recipe*
  - Instance Variables:
    - *name*: (string) the name of the recipe
    - *ingredients*: (list) the ingredients required for the recipe.
    - *prep_time_minutes*: (int) the preparation time in minutes.
    - *cook_time_minutes*: (int) the cooking time in minutes.
  - Methods
    - *__init__*
      - The constructor
- *Cook*
  - Instance Variables:
    - *name*: (string) name of the cook.
    - *recipes_list:* (list) a private list to store *Recipe* objects that the cook has learned. Directly assign it to hold an empty list. Make sure you make the variable have a private scope
  - Methods
    - *__init__*
      - The constructor.
    - *learn_recipe*
      - adds a recipe to the cook's list of recipes.
    - *display_recipes*
      - prints out information about all the recipes the cook knows.
- *ExpertCook (*Inherits from Cook)
  - Instance variables:
    - Everything from *Cook* through the use of *super()*
    - *training_location:* (string) where the expert cook trained

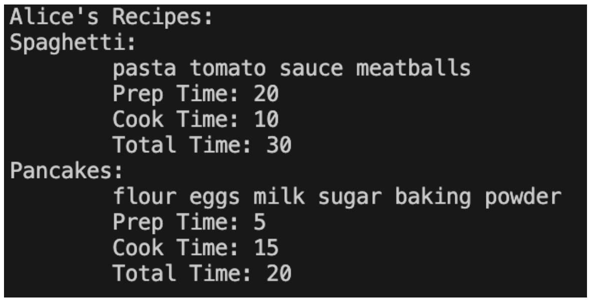
  - Methods:
    - *display_recipes*
      - prints out information about all the recipes the cook knows, except that expert cooks have the displayed prep time of the recipe decreased by half.

Logical Flow:

- Make a bunch of recipe objects according to the description of the *Recipe* class constructor given above. *Recipe* objects each have a name, a list of ingredients, a prep time in minutes, and a cook time in minutes.
    - You can make whatever *Recipe* object you want. To save you time typing, here's a few if you want to paste some in, but it is up to you.

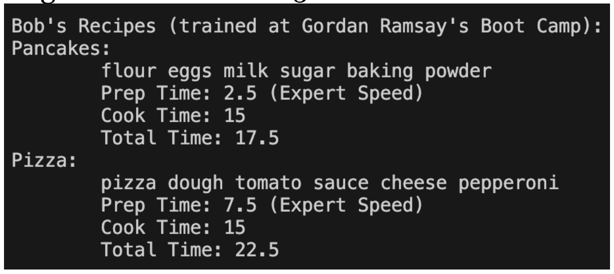
```python
spaghetti = Recipe("Spaghetti", ["pasta", "tomato sauce", "meatballs"], 20, 10)
salad = Recipe("Salad", ["lettuce", "tomato", "cucumber", "salad dressing"], 10, 0)  # Assuming no cook time
pizza = Recipe("Pizza", ["pizza dough", "tomato sauce", "cheese", "pepperoni"], 15, 15)
chicken_curry = Recipe("Chicken Curry", ["chicken", "curry powder", "coconut milk", "rice"], 20, 25)
pancakes = Recipe("Pancakes", ["flour", "eggs", "milk", "sugar", "baking powder"], 5, 15)
chocolate_cake = Recipe("Chocolate Cake", ["flour", "cocoa powder", "eggs", "sugar", "butter"], 20, 30)
beef_stew = Recipe("Beef Stew", ["beef", "potatoes", "carrots", "onions", "beef broth"], 15, 105)
```

- You should then make at least 1 *Cook* object and at least 1 *ExpertCook* object according to the constructor toward the beginning of these instructions above. *ExpertCook* objects have the same instance variables as regular *Cook* objects, but should also have a *training_location* instance variable, which is a string describing where they trained.
- Make it so every *Cook* and *ExpertCook* object stores 2 random *Recipe* objects inside of their *recipes* list instance variable. This should happen through the *learn_recipe* method:
    - When *learn_recipe* is called, it should take in a *Recipe* object as an argument and add the *Recipe* object to the *Cook/ExpertCook* that called the method. But, it should only add the *Recipe* if it isn't already in that *Cook/ExpertCook's* recipe list.
    - If a Cook/Expert cook tries to learn a recipe they already know, it shouldn't add anything to their recipes list and instead print out the message
        - "You already know how to cook <recipe name>!"
- Once every *Cook/ExpertCook* has 2 *Recipes* in their recipe list, run the *display_recipes* method on every *Cook/ExpertCook* object.
    - *display_recipes*
        - If the *Cook/ExpertCook* doesn't know any recipes, then it should print out "<cook name> knows no recipes."
        - Otherwise, for every recipe in the *Cook's recipes* list, it should print out the recipe name, all the ingredients in the recipe, the prep time for the recipe, the cook time for the recipe, and the total time for the recipe (the total of the prep and cook time)

- 

```
Alice's Recipes:
Spaghetti:
        pasta tomato sauce meatballs
        Prep Time: 20
        Cook Time: 10
        Total Time: 30
Pancakes:
        flour eggs milk sugar baking powder
        Prep Time: 5
        Cook Time: 15
        Total Time: 20
```

  - When an *ExpertCook* object runs this method, it should do the same thing except it should print out “(trained at <training location>)” next to their name, display the prep time as half of what the recipe states, and also print out “(Expert speed)” next to the prep time.
  - For example, below see how when an expert preps Pancakes, the 5-minute prep time gets decreased to 2.5 minutes:

- 

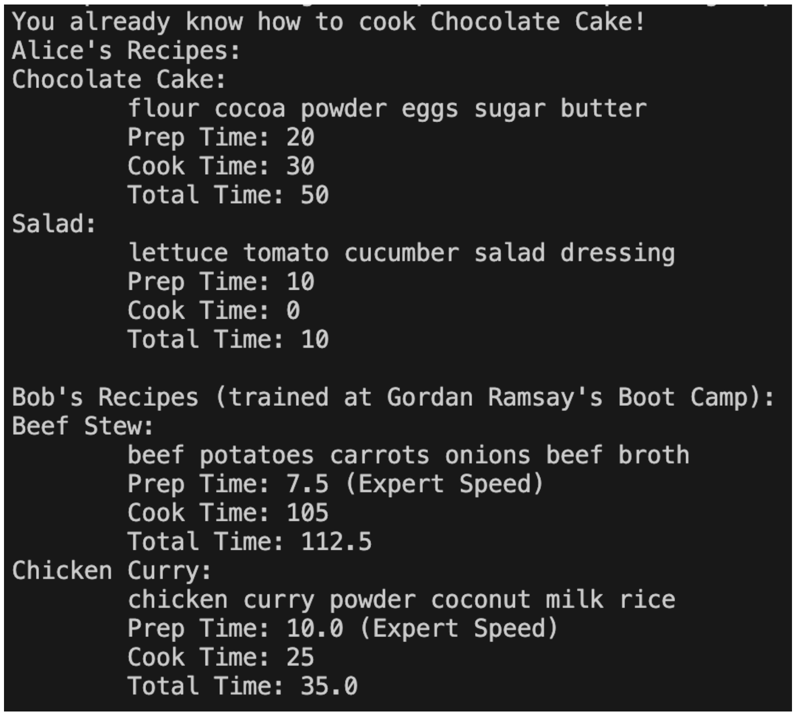
```
Bob's Recipes (trained at Gordan Ramsay's Boot Camp):
Pancakes:
        flour eggs milk sugar baking powder
        Prep Time: 2.5 (Expert Speed)
        Cook Time: 15
        Total Time: 17.5
Pizza:
        pizza dough tomato sauce cheese pepperoni
        Prep Time: 7.5 (Expert Speed)
        Cook Time: 15
        Total Time: 22.5
```

None of the methods can reference global variables.

You are not required to catch exceptions.

Example Output:

Note that in this output, by random chance the *learn_recipe* function was called with a recipe that had already been learned, so the message “You already know how to cook X!” printed out when I ran it.



```
You already know how to cook Chocolate Cake!
Alice's Recipes:
Chocolate Cake:
        flour cocoa powder eggs sugar butter
        Prep Time: 20
        Cook Time: 30
        Total Time: 50
Salad:
        lettuce tomato cucumber salad dressing
        Prep Time: 10
        Cook Time: 0
        Total Time: 10

Bob's Recipes (trained at Gordan Ramsay's Boot Camp):
Beef Stew:
        beef potatoes carrots onions beef broth
        Prep Time: 7.5 (Expert Speed)
        Cook Time: 105
        Total Time: 112.5
Chicken Curry:
        chicken curry powder coconut milk rice
        Prep Time: 10.0 (Expert Speed)
        Cook Time: 25
        Total Time: 35.0
```

Rubric:

Since this is a practice problem, I didn't create a grading rubric, but you can check your work against the example solution in the repository. You could also paste the instructions and your code into an AI and ask it if you did anything wrong.