Object-Oriented Programming
50:198:113 (Spring 2022)

Homework:	**5**	**Professor:**	**Suneeta Ramaswami**
Due Date:	**4/13/22**	**E-mail:**	`suneeta.ramaswami@rutgers.edu`
Office:	**321 BSB**	**URL:**	`http://crab.rutgers.edu/~rsuneeta`
		Phone:	**(856)-225-6439**

Homework Assignment 5

The assignment is due by 11:59PM of the due date. The point value is indicated in square braces next to each problem. Each solution must be the student's own work. Assistance should only be sought or accepted from the course instructor. Any violation of this rule will be dealt with harshly.

This assignment requires you to go further with classes. In Problem 1, you are asked to implement a new class called `Trip` that has a `Date` object as an instance attribute. In Problem 2, you are asked to implement a class called `TripSchedule`, which is a collection of `Trip` objects.

As usual, you are graded not only on the correctness of the code, but also on clarity and readability. I will deduct points for not following the guidelines for your class design, poor indentation, poor choice of object names, and lack of documentation. You are expected to provide docstring documentation for modules, classes and their methods, as well as functions.

Problem 1 [30 points] Trip. In a company, it is frequently necessary to keep track of the travel schedule for an employee (henceforth referred to as a person) whose job requires frequent travel. In this problem, you are asked to implement a class called `Trip`, described below, that will help the company with this task. Create a module called `trip.py` to contain this class. The module `test_trip.py` is provided for you to test your implementation of the `Trip` class. In the next problem, you will use the `Trip` class to implement a container class to hold multiple trips for an employee.

Important: You will need `Date` instances to implement this class. I am providing you with my implementation of the `Date` class in the module `date.py`. Please use this module for your implementation of the `Trip` class. Do not use your own `Date` class as it will allow me to carry out standardized testing of your work on this homework assignment. Make sure that you include the line `from date import Date` at the top of your `trip.py` module.

Details about the `Trip` class implementation are as follows. An instance of the `Trip` class allows us to store and retrieve information about a specific trip. The attributes of a trip are as follows, which should all be *private*:

- `destination`, the destination city of the trip (this is a string). To keep things simple, we'll assume the person travels to only one city at a time,
- `depdate`, the date on which the person departs on the trip (this is a `Date` object),
- `duration`, the duration of the trip; that is, *the number of nights the person is away from the home town* (this is an integer ≥ 1).

We include various methods that manipulate attributes of this class, as described below.

1. A constructor to initialize the three trip instance attributes described above. Keep in mind that `depdate` is a `Date` instance.

2. A method `setDestination` to set the trip destination to a given value (a string).
3. A method `setDeparture` to set the trip departure date to a given value (a `Date`).
4. A method `setDuration` to set the trip duration to a given value (an integer).
5. A method `destination` that returns the destination of the trip.
6. A method `departure` that returns the departure date of the trip.
7. A method called `duration` that returns the duration of the trip.
8. A method `arrival` that returns the arrival date for the trip (the date on which the person arrives back to the home town). Note that the return value is a `Date`. *Use* `Date` *methods to implement this function. Do not repeat code needlessly.*
9. A method called `overlaps` with two parameters `self` and `other`, where `other` is also a `Trip`. The method returns `True` if the trips `self` and `other` overlap. Two trips are considered to overlap if the dates of travel (including departure and arrival dates) of one overlap with the dates of travel of the other.
10. A method called `containsweekend` that returns `True` if the trip contains at least one day of a weekend (Saturday or Sunday) and False otherwise. For example, if a trip `T` to London was 2 days long and started on March 26, 2015, then `T.containsweekend()` will return `True`. However, if `T` started on March 24, 2015, then `T.containsweekend()` will return `False`.
11. A method `__str__` to print the trip details in a neatly formatted way. The trip details include the destination, the duration of the trip, the departure date (indicate the day of week as well), and the arrival date (again, indicate the day of week). Here is an example output:

```
Destination: Paris
Duration: 6 days
Departure: Sunday, April 17, 2022
Arrival: Saturday, April 23, 2022
```

12. A method `__repr__` to return a suitable string representation of a trip.

Problem 2 [50 points] TripSchedule and TripScheduleIterator. You will implement these classes in a module called `tripschedule.py`. This class requires you to use the `Trip` class from Problem 1. Make sure that you insert the following line at the top of your `tripschedule.py` module:

```
from trip import Trip
```

The module `test_tripschedule.py` is provided for you to test your implementation of the `TripSchedule` class.

An instance of the `TripSchedule` class stores a collection of trips that form the trip schedule for one person. Note that the trips must conform to the following consistency requirement: no two trips can conflict with one another. This means that the dates on which an employee is traveling on one trip cannot overlap with the dates on which s/he is traveling on another. We also do not allow a person to depart on a trip on the same day s/he arrives back from another.

The `TripSchedule` class will be a "container" for other objects (in particular, for `Trip` objects). You are required to include the following methods for this class:

1. A constructor that creates an empty trip schedule. Store the trips in the schedule in a list.
2. A method called `insert` to add a new trip to the schedule *if it does not conflict with existing ones.* If there is a conflict, print an appropriate message and raise an exception. Use `Trip` methods to implement this function. Points will be deducted for code redundancy.
3. A method called `delete` to delete a trip from the schedule.
4. A method called `__len__` (this overloads the built-in `len()` function) to return the length of the trip schedule (i.e., the total number of trips in the schedule).
5. A method called `__getitem__` to overload the index operator. An index value of `j` returns the `j`-th trip in the schedule (use the index 0 for the first trip, 1 for the second trip, and so on...)
6. A method called `__iter__` to create an iterator for a trip schedule. This method should return a new tripschedule iterator object.
7. A method called `search` to search the schedule by a keyword that can be either a destination or a month. Therefore, this method has one parameter, `keyword`.) If the keyword is an integer in the range 1 to 12 (inclusive), then all trips in the schedule that start in that month should be printed out. Otherwise, the keyword is assumed to be a destination and all trips in the schedule with that destination should be printed out. Use the `print` function for `Trip`s for proper display. Also print the trips in sorted order of departure date.
8. A method called `available` with two parameters, `month` and `year`, to search the schedule for all available dates in `month` (an integer between 1 and 12) of `year`. Available dates are dates on which there is no travel scheduled. The function **returns** a list of all available dates in `month` of `year`. *Note* that this method returns a list of `Date` instances.
9. A method called `weekend_travel` with one parameter, `yr`, to search the schedule for all trips in year `yr` that involve weekend travel. The function **returns** a list of all such trips. The list must store the trips in sorted order of departure date. *Note* that this method returns a list of `Trip` instances.
10. A method called `earliest`, which returns the trip in the schedule that has the earliest departure date of all the trips. *Note* that this method returns a `Trip` instance.
11. A method called `last`, which returns the trip in the schedule that has the latest departure date of all the trips. *Note* that this method returns a `Trip` instance.
12. A method called `sortbydeparture`, which sorts all the trips in the schedule by their departure dates. After this method is called, the instance attribute list storing all the trips in the schedule should be sorted.
13. A method `__str__` to return a string representation of the trip schedule. *Hint:* Use the `str` function for `Trip` objects for a straightforward implementation of this method.
14. A method `__repr__` to return a suitable string representation of a trip schedule.

An instance of `TripScheduleIterator` is an iterator for a `TripSchedule`. This class will have exactly two methods:

1. `__init__`: You must decide what parameters to pass to the constructor and which instance attribtues to create for a tripschedule iterator.

2. `__next__`: This method returns the next trip in the tripschedule.

Submission Guidelines

Implement the first problem in a module called `trip.py` and the second problem in a module called `tripschedule.py`. *Your name and RUID should appear as a comment at the very top of each module.* Points will be deducted if you do not follow the specified naming convention.

Test each of your programs thoroughly before submitting your homework. When you are ready to submit, upload your files on Canvas as follows:

1. Go to the "Assignments" tab of the Canvas site for this course.
2. Click on "Programming Assignment #5" under Homework Assignments.
3. Upload your homework files (`trip.py` and `tripschedule.py`) when you are ready to submit.

You must submit your assignment at or before 11:59PM on April 13, 2022.