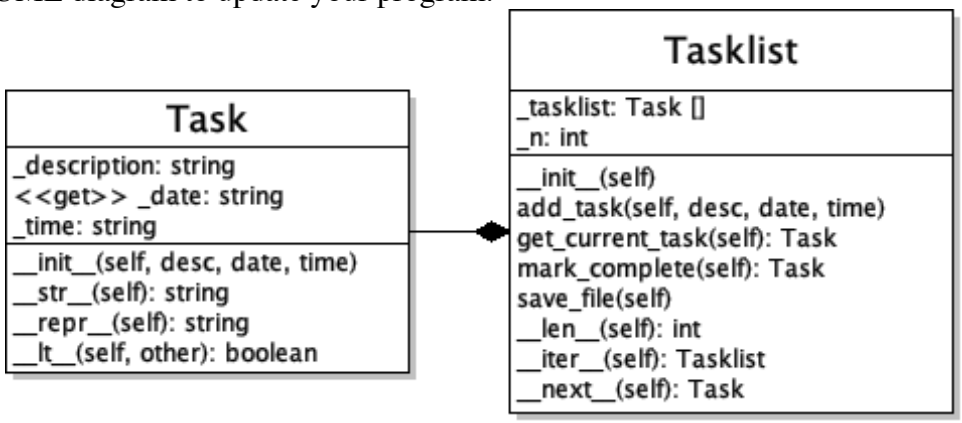
CECS 277 – Lab 13 – Iterator

Task List – Part 2

Update Lab 6 to use an iterator to display the contents of the task list. The user should be able to do everything that they were able to do before, plus the added ability to search by date. Use the following UML diagram to update your program:



Classes:

1. Task:
    a. Update each of the attributes to have a leading underscore.
    b. All methods are the same as before, just add a property to access the date.
2. Tasklist:
    a. Attributes: tasklist – stores the list of task objects (update it so that it has a leading underscore), n – counter for iterator.
    b. __init__, add_task, mark_complete, save_file, and len – no changes. Do not initialize the attribute n in the __init__.
    c. get_current_task(self) – return the first task object.
    d. __iter__(self) – initialize the iterator attribute (n) and return self.
    e. __next__(self) – iterate the iterator one position at a time. Raise a StopIteration when the iterator reaches the end of the tasklist, otherwise return the Task object at the iterator's current position.

Main file ('main.py') – create the following functions:

1. main_menu() – update the menu to have the additional option to 'Search by Date'.
2. get_date() and get_time – no changes.

Construct a Tasklist object and then repeatedly display the number of tasks and then prompt the user to choose from the following options until they quit the program:

1. Display all tasks – displays a numbered list of all the tasks in sorted order. It should be using the iterator that you created in the Tasklist class.
2. Display current task – displays the first task using Tasklist's get_current_task method. If there are no tasks, then display a message that says all their tasks are complete.
3. Mark current task complete – Displays the current task, removes it, and then displays the new current task by calling Tasklist's mark_complete method. If there are no tasks, then display a message.

4. Add new task – Prompts the user to enter a new task description, due date, and time, and adds the new task to the list by calling the Tasklist's add_task method.
5. Search by date – Prompts the user to enter a date (MM/DD/YYYY). Use the iterator to loop through the list, if the task's date is the same as the user's, then display the task. It should display all tasks with the matching date (not just the first one).
6. Save and quit – saves the updated contents of the list back to the file (does not append) and then quits the program.

Example Output:

```
-Tasklist-
Tasks to complete: 7
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
6. Save and quit
Enter choice: 4
Enter a task: Buy Flowers
Enter due date:
Enter month: 11
Enter day: 15
Enter year: 2023
Enter time:
Enter hour: 2
Enter minute: 00

-Tasklist-
Tasks to complete: 8
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
6. Save and quit
Enter choice: 1
Current task is:
Do Math Homework - Due:
11/05/2023 at 12:00

-Tasklist-
Tasks to complete: 8
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
6. Save and quit
Enter choice: 3
Marking current task as
complete:
Do Math Homework - Due:
11/05/2023 at 12:00
New current task is:
Write Email to Supervisor -
Due: 11/13/2023 at 12:00

-Tasklist-
Tasks to complete: 7
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
6. Save and quit
Enter choice: 5
Enter date to search:
Enter month: 11
Enter day: 15
Enter year: 2023
Tasks due on 11/15/2023:
1. Buy Flowers - Due:
11/15/2023 at 02:00
2. Do English Assignment -
Due: 11/15/2023 at 12:00
3. Babysit for Neighbor Kids
- Due: 11/15/2023 at 20:00

-Tasklist-
Tasks to complete: 7
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
```

```
6. Save and quit
Enter choice: 2
Tasks:
1. Write Email to Supervisor
- Due: 11/13/2023 at 12:00
2. Buy Flowers - Due:
11/15/2023 at 02:00
3. Do English Assignment -
Due: 11/15/2023 at 12:00
4. Babysit for Neighbor Kids
- Due: 11/15/2023 at 20:00
5. Haircut - Due: 11/20/2023
at 15:00
6. Do Grocery Shopping - Due:
11/21/2023 at 12:00
7. Clean House - Due:
11/26/2023 at 12:00

-Tasklist-
Tasks to complete: 7
1. Display current task
2. Display all tasks
3. Mark current task complete
4. Add new task
5. Search by date
6. Save and quit
Enter choice: 6
Saving List...
```

Notes:

1. Please place your name, date, and a brief description of the program in a comment block at the top of your program (main.py).
2. Use the check_input module to check the user's input for invalid values.
3. Do not add any additional attributes or methods to the classes.
4. Please do not use the datetime date or time classes.
5. Do not modify the methods or functions defined above (ie. same name and parameters).
6. You may create additional functions in main.py as needed.
7. Do not create any global variables or use attributes directly.
8. Use docstrings to document your class, its attributes, methods, and main functions. Add brief comments in your program to describe sections of code.
9. Thoroughly test your program before submitting:
    a. Make sure the file is read in properly, tasks are constructed and stored in the list.
    b. Make sure that the tasks are sorted in ascending order. The __lt__ method should sort by year, and if those are the same, it should sort by month, then day, then hour, then minute, then if all of those are the same, then sort the description.
    c. Make sure that it displays all the tasks in sorted order and uses the iterator.
    d. Make sure that you display the total number of tasks to complete.
    e. Make sure that the current task is the soonest due date and time.
    f. Make sure that the current task is removed when marking it complete.
    g. Make sure that the program does not crash when viewing, completing, iterating through, or saving an empty (or near empty) task list.
    h. Make sure to error check all user input: Main menu: 1-6, Year: 2000-2100, Month: 1-12, Day: 1-31, Hour: 0-23, Minute: 0-59.
    i. Make sure the list is resorted after adding a task.
    j. Make sure that you write to the file in the same format (ie. comma separated with no spaces after the commas (spaces are ok in the description)).
    k. Avoid using commas when entering a description (I won't specifically test for this, but it will mess up your file).

Tasklist Part 2 – Time estimate: 3 hours

Tasklist Part 2 **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Task Class (separate file)**:** 1. Updated attributes with leading underscore. 2. Added property for date value. 3. Other methods are the same.					
Tasklist Class (separate file)**:** 1. Updated attribute with leading underscore. 2. Removed getitem method. 3. Added get_current_task method. 4. Other methods are the same.					
Added Iterator: 1. Added iter and next methods to Tasklist class. 2. iter initializes n (not initialized in main) and returns the iterator. 3. next iterates through the tasklist one item at a time and returns the task. 4. Correctly raises StopIteration exception when it reaches the end of the list.					
Main file (in separate file)**:** 1. Main creates a Tasklist object. 2. Prompts user with menu 1-6. 3. Options 2 and 5 use the iterator. 4. Option 5 prompts user for date and displays all matches. 5. Updated to use get_current_task instead of square brackets. 6. Program otherwise works as it did before.					
Code Formatting: 1. Correct documentation. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables or accessing attributes directly. 5. Correct spacing.					