Consider a digital personal assistant similar to Alexa and Siri that could manage your schedule. How would you design a tool such that it can efficiently find events during a certain period of time (without the speech recognition/generation component)?

The goal of HW5 is to allow the user to specify a time range to get the corresponding events. Also, we would like the user to be able to add and cancel events as the user’s schedule changes. To improve efficiency, your implementation uses a SkipListMap class that has at least the following operations:

- \( \text{get}(k) \) \( \text{[p. 403; pseudocode is on p. 439]} \)
- \( \text{put}(k, v) \) \( \text{[p. 403; pseudocode is on p. 440]} \)
- \( \text{remove}(k) \) \( \text{[p. 403; p. 441]} \)
- \( \text{subMap}(k_1, k_2) \) \( \text{[p. 428]} \)

Use DoublyLinkedList \( \text{[p. 135]}, \) which you can modify, to implement the SkipListMap class. You may also implement your own DoublyLinkedList. Use FakeRandomHeight for \( \text{put}(k, v) \) (to facilitate easier debugging and testing). Program files for DoublyLinkedList and FakeRandomHeight are on the course website.

**Input:** The command-line argument for HW5.java is the name of a file, which has one of the following actions on each line:

- AddEvent time event
- CancelEvent time
- GetEvent time
- GetEventsBetweenTimes startTime endTime
- GetEventsForOneDay date
- GetEventsForTheRestOfTheDay currentTime
- GetEventsFromEarlierInTheDay currentTime
- PrintSkipList

For simplicity, time is an integer in MMDDHH format and date is in MMDD format (MM is 01-12, DD is 01-31, and HH is 00-23) \( \text{[leading zeros are optional]} \). The timestamps are unique and each timestamp can only have at most one event. Sample input is on the course website.

**Output:** Output goes to the standard output (screen), each line has a result for the corresponding action:

- AddEvent time event \( \text{[ExistingEventError: existingEvent]} \)
- CancelEvent time/event NoEventError
- GetEvent time/event none
- GetEventsBetweenTimes startTime endTime time1 event1 ... or none
- GetEventsForOneDay date time1 event1 ... or none
- GetEventsForTheRestOfTheDay currentTime time1 event1 ... or none
- GetEventsFromEarlierInTheDay currentTime time1 event1 ... or none
- PrintSkipList
  (Sh) empty
  (S1) time1 event1 ...
  (S0) time1 event1 ...

Sample output is on the course website.
Extra Credit (10 more points) To allow more interesting actions, instead of using int/Integer for timestamps in the Skip List, use the GregorianCalendar class \(^1\) (assume year is 2017), which extends the Calendar class \(^2\). The time/date format for input and output remains the same. Also, allow these *additional* input actions:

- AddWeeklyEvent timeOfFirstEvent numOfWeeks event
- GetEventsForTomorrow currentTime
- GetEventsForTheComingWeekday currentTime Mon/.../Sun
- GetEventsForTheComingWeekend currentTime

and output:

- AddWeeklyEvent timeOfFirstEvent numOfWeeks event [ExistingEventError:existingEvent]
- GetEventsForTomorrow currentTime time: event1 ... or none
- GetEventsForTheComingWeekday currentTime Mon/.../Sun time: event1 ... or none
- GetEventsForTheComingWeekend currentTime time: event1 ... or none

If currentTime is during a weekend (Sat/Sun), the coming weekend is the next weekend.

Submission: Submit HW5.java that has the main method, SkipListMap.java, (modified) DoublyLinkedList.java, FakeRandomHeight.java and other program files. Submissions for Individual and GroupHelp have the same guidelines as HW1. For Extra Credit, submit HW5Extra.java, SkipListMapExtra.java, and related program files. Note the late penalty on the syllabus if you submit after the due date and time as specified at the top of the assignment.

\(^1\) https://docs.oracle.com/javase/8/docs/api/java/util/GregorianCalendar.html
\(^2\) https://docs.oracle.com/javase/8/docs/api/java/util/Calendar.html