

## Background

A common activity that many of us perform everyday is filling out our schedule. This program will be a simple electronic scheduler. You will be able to enter events to occur on a days June, 2003, and view the scheduled events. The events will start and end at different times and will start and end on an hour. Your program should schedule the event only if it does not overlap with a previously scheduled event.

## Details

In this project, you will only schedule events for June, 2003. The days in which you are scheduling specified in the commands. The primary issue is whether events overlap or not. If an event does not overlap with a previously scheduled event, then it is allowed to be scheduled. Each event is guaranteed to last an integral number of hours from 1 to 3. In addition, no event will last past the end of a day. The commands for this project are as follows:

*add*<tab><Day><tab><Time><tab><Duration><tab><Event Description>

The add command attempts to schedule an event on the specified date at the specified time. If another event has already been scheduled, the event should not be scheduled.

*remove*<tab><Day><tab><Time>

The remove command deletes a scheduled event on the specified date at the specified time. If no event starts at that date and time, then the command is ignored, except for printing the command to the output file..

*display*<tab><Day>

The display command will result in information about all of the scheduled events on the given date. If the specified date is the keyword "All" rather than a valid date, then the entire month is printed. Before the month is printed the name of the month is printed. Also, before the dates for the month are printed, column headings consisting of the first three letters for each day of the week should be printed across the top. For more details, see the sample output.

If the day is not the keyword "All", then the particular day is printed. If there are events scheduled for that day, then the information, as shown in the sample output file, should be output. Note that the events are output in scheduled order, and events are numbered.

## Input

All input commands will be in a script file called "Calendar.ini". After each command is read, the entire command, including all arguments must be printed to the output file. You are guaranteed that all input is valid. Here are some other basic definitions:

- Day is any number greater than zero and less than 31, or the sentinel work "ALL".
- Time is any hour (0-23) and minutes (00-59) represented in military time with a colon separating the hour from the minutes.
- Duration is how long the event lasts in minutes and will be 60, 120, or 180.
- Event Description is a string describing the event

Below is a sample input file:

```

add 11 3:00 180 My Birthday
display 11
add 10 0:00 180 Kelly's Birthday
display 10
add 9 2:00 60 Party
display 9
remove 10 0:00
display 10
add 10 2:00 60 Party
display 10
add 12 15:00 120 Watch TV
display 12
add 12 12:00 60 Get my picture taken
display 12
remove 11 3:00
display 11
add 11 12:00 120 Get my picture taken
display All

```

### Output

Below is the output, called "Calendar.txt" for the sample input given above. Be sure to pay close attention to capitalization and punctuation in the output since the Curator tends to be very fussy:

```

Programmer: Rich Wheaton
CS 1044 Event Scheduling
~~~~~
Add:      June 11
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Display:  June 11
~~~~~
Event: 1
June 11
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Add:      June 10
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
~~~~~

```

```
Display:      June 10
~~~~~
Event: 1
June 10
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
~~~~~
Add:          June 9
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Display:      June 9
~~~~~
Event: 1
June 9
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Remove:       June 10 at 0:00
~~~~~
Display:      June 10
~~~~~
~~~~~
Add:          June 10
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Display:      June 10
~~~~~
Event: 1
June 10
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Add:          June 12
Event at 15:00 for 120 minutes
Event Description: Watch TV
~~~~~
```

```

Display:      June 12
~~~~~
Event: 1
June 12
Event at 15:00 for 120 minutes
Event Description: Watch TV
~~~~~
Add:          June 12
Event at 12:00 for 60 minutes
Event Description: Get my picture taken
~~~~~
Display:      June 12
~~~~~
Event: 1
June 12
Event at 12:00 for 60 minutes
Event Description: Get my picture taken
Event: 2
June 12
Event at 15:00 for 120 minutes
Event Description: Watch TV
~~~~~
Remove:       June 11 at 3:00
~~~~~
Display:      June 11
~~~~~
~~~~~
Add:          June 11
Event at 12:00 for 120 minutes
Event Description: Get my picture taken
~~~~~
Display:      All
~~~~~
~~~~~
                June
Sun Mon Tue Wed Thu Fri Sat
  1  2  3  4  5  6  7
  8  X  X  X  X 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
~~~~~

```

### Implementation Details

The main data structure for this program is a two-dimensional array of the structure that you define to hold the data. The only other specific requirements are as follows:

- Use of functions where appropriate. You must use at least 4.
- All functions must be commented
- As described above, you must use an array of structs.
- You must follow all the Standards of Good Programming Practices.

### Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically.

Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and submission link can be found at:

<http://www.cs.vt.edu/curator/>

**Pledge:**

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

**Failure to include this pledge in a submission is a violation of the Honor Code.**