

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 given day 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 schedule events for one day. The day in which you are scheduling is not important. The 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 the day. The commands for this project are as follows:

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

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

remove<tab><Time>

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

display

The display command will result in information about all of the scheduled events being printed. If there are no scheduled events, the command should be ignored, except for printing the command to the output file.

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. Below is a sample input file:

```
add 3:00 180 My Birthday
display
add 0:00 180 Kelly's Birthday
display
add 2:00 60 Party
display
remove 0:00
display
add 2:00 60 Party
display
```

Output

Below is the output 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 Overlapping
~~~~~
Add:
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Display:
~~~~~
Event: 1
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Add:
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
~~~~~
Display:
~~~~~
Event: 1
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
Event: 2
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Add:
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Display:
~~~~~
Event: 1
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
Event: 2
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Remove: 0:00
~~~~~

```

```

Display:
~~~~~
Event: 1
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~
Add:
Event at 2:00 for 60 minutes
Event Description: Party
~~~~~
Display:
~~~~~
Event: 1
Event at 2:00 for 60 minutes
Event Description: Party
Event: 2
Event at 3:00 for 180 minutes
Event Description: My Birthday
~~~~~

```

Implementation Details

The main data structure for this program is an array of the structure that you define to hold the data. This array will be of size 24; each cell is to represent an hour. The only other specific requirements are as follows:

- Use of functions where appropriate. You should use at least 4.
- You must use a switch statement.
- As describe 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.