Background
This project is similar to the last project in that we will be scheduling events; however in this project we will only be scheduling events on one day. We will take events that start and end at different times and scheduled them on an hourly basis only if they do not overlap with an already scheduled event.

Details
In this project you will schedule events for one day. The day in which you are scheduling is not important to this project. The issue is whether the events overlap or not. If an event does not overlap with an already scheduled event, then it is allowed to be scheduled. If an event does overlap with an already scheduled event, then it is not scheduled. Each event is guaranteed to last an integral number of hours from 1 to 3.

The commands will be the same as the last project, with the exception that there will be no display All command, since we are only dealing with 1 day. The display a day command output will be altered to display each scheduled event. In this project the time and duration will be important to scheduling each event. It is your job to check to make sure that each event does not overlap with any other scheduled event. If an event lasts for 3 hours and begins at noon, then no other event change start or end between noon and 3; however other events can end at noon or begin at three. The duration will still be in minutes, but will be any of the following values: 60, 120, or 180. Because the fact that the events are integral number of hours you will use an array of 24 cells to represent the day. You should only have one array.

The input and output files will retain the same names and structure from the last project.

To represent an event you will use a struct. The struct must have a field for each of the items given in the input. The display for a day will be altered to only display the events that have been scheduled for that day. Any hours that do not have a scheduled event should not be displayed and the events must be displayed in time order.

Below is a sample input file and its corresponding output file.

```
add 11 3:00 180 My Birthday
display 11
add 11 0:00 180 Kelly's Birthday
display 11
add 11 2:00 60 Party
display 11
remove 11 0:00
display 11
add 11 2:00 60 Party
display 11
```

Programmer: David McPherson
CS 1044 Event Overlapping
Add: November 11
Event at 3:00 for 180 minutes
Event Description: My Birthday

Display: November 11

Event: 1
November 11
Event at 3:00 for 180 minutes
Event Description: My Birthday

Add: November 11
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday

Display: November 11

Event: 1
November 11
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
Event: 2
November 11
Event at 3:00 for 180 minutes
Event Description: My Birthday

Add: November 11
Event at 2:00 for 60 minutes
Event Description: Party

Display: November 11

Event: 1
November 11
Event at 0:00 for 180 minutes
Event Description: Kelly's Birthday
Event: 2
November 11
Event at 3:00 for 180 minutes
Event Description: My Birthday

Remove: November 11 at 0:00

Display: November 11

Event: 1
November 11
Event at 3:00 for 180 minutes
Event Description: My Birthday

Add: November 11
Event at 2:00 for 60 minutes
Event Description: Party

Display: November 11
The output is extremely similar to the last project with a few minor punctuation changes. Make sure that you are correct in your output.

Implementation Details

The main data structure for this project is an array of the structure you create to hold the data. This array will be of size 24; each cell is to represent an hour. There are no other explicit constraints for this project except the following:

- Use of functions where appropriate. You can most likely have at least 3.
- You must use a switch statement.
- You must use an array of structs.
- You should not have more than one array.
- You must follow all the Standards of Good Programming Practices.

Submitting your project

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:

```c++
// 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.
```
Failure to include this pledge in a submission is a violation of the Honor Code.