ECS 40 **Project #2 (50 points)** **Spring 2013**

File name: calendar.cpp
Due: 2nd May 2013 at 11:55 pm
Gradebot will be used for submission. The appts.csv and input.csv files are for you to know the file formats. When you use gradebot, you ONLY have to submit your program calendar.cpp. Gradebot will then run your program using these files (they are already uploaded on the gradebot server).

This will be the first of a series of programming assignments to create an appointment calendar program. For this program, you will write a program to read in the information from the file appts.csv, and search for appointments based on date or subject. The information should be stored in an array named days.

Ideally, the body of any function cannot be more than 30 lines long. The output of your program must be identical to that of mine. Your program should compile with no warnings. Though the name of the file ends with .cpp, you may write your program in either C or C++. In either case, you will use the g++ compiler.

Specifications:

1. appts.csv is a "comma separated value" file of indeterminate size.
   1.1. The first line contains the title of each column.
   1.2. Each succeeding line contains the information for an appointment.
   1.3. You should use **strtok()** available in string.h to parse each line. I also found atoi(), strlen(), strcpy, and strchr() useful.
2. days array is a dynamically allocated of typedef structs Day.
   2.1. It is sorted by the date of each Day, with initial size of 30.
   2.2. Whenever the array needs to be resized, it should be doubled in size. Note that this will happen more than once!
      2.2.1.I suggest that your readFile() function return a pointer to the array, and have the array's size as a parameter.
   2.3. Deallocate the array at the end of the program.
3. Typedef struct Day contains a short named "day", a short named "month", an array named "appts" of eight pointers to typedef struct Appointment, and a short named "apptCount".
4. Typedef struct Appointment contains: "startTime" and "endTime" of typedef struct Time, a dynamically allocated array of char named "subject", and a dynamically allocated array of char named "location".
   4.1. Deallocate the subject and location at the end of the program.
5. Typedef struct Time uses contains a short named "hour" (0 – 23), a short named "minute".
6. The test cases are provided in input.csv. Each line of the file is a test cases. And each test case has the following format:**Option, value**
   There are only two valid options for this program. Option = 1 means that we want to search the calendar for all the appointments at a particular date.The **value** field will then contain that particular date. An example of such a test case:

   1,13/12

   In the above test case, we want to search for all the appointments at 13/12. Option = 2 means that we want to search the calendar for appointments for a particular topic/subject. The **value** field will then contain that particular topic/subject. An example of such a test case:

   2, ECS40

   Here, we are trying to find all the appointments which have the topic ‘ECS40’.

Your program should do range checking for the following:

1. The option.
2. The date.

```
$ ./calendar.out appts.csv input.csv
CSV Entry- 1)   1,13/12

Month and day (mm/dd) >> 13/12
13/12 is not a valid date.
CSV Entry- 2)   2,ECS 30

ECS 30 was not found as a subject in the calendar.
CSV Entry- 3)   1,12/12

Month and day (mm/dd) >> 12/12
Start End   Subject         Location
10:30 12:30 ECS 30-D Final  26 Wellman
13:30 17:00 Office Hours    3052 Kemper
CSV Entry- 4)   1,invDate

Month and day (mm/dd) >> invDate
invDate is not a valid date.
CSV Entry- 5)   2,ECS  30

ECS  30 was not found as a subject in the
calendar.
CSV Entry- 6)   3,wrong input

Invalid Choice
CSV Entry- 7)   2,ECS 30-D Final

Date  Start End   Subject         Location
12/12 10:30 12:30 ECS 30-D Final  26 Wellman
```