Calendar Tool with Automated Testing

The Gregorian calendar (the one we use) consists of the following 12 months ([https://www.timeanddate.com/calendar/months/)Links to an external site.](https://www.timeanddate.com/calendar/months/)):

January - 31 days  
February - 28 days in a common year and 29 days in leap years  
March - 31 days  
April - 30 days  
May - 31 days  
June - 30 days  
July - 31 days  
August - 31 days  
September - 30 days  
October - 31 days  
November - 30 days  
December - 31 days

Determining if a year is a leap year is based on the following criteria ([https://www.timeanddate.com/date/leapyear.html)Links to an external site.](https://www.timeanddate.com/date/leapyear.html)):

The year must be evenly divisible by 4;  
If the year can also be evenly divided by 100, it is not a leap year;  
unless... the year is also evenly divisible by 400. Then it is a leap year.  
  
According to these rules, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300, and 2500 are not leap years.

**Problem:**

Repeatedly prompt a user to enter a year and month, output to the user the number of days in that month.

You must break up the problem into functions, for example, there should be at least one function to take user input, one function to determine if a users input is valid, one function to calculate the number of days in a month and one function to determine if a year is a leap year. That is a minimum of 4 functions. There should be very little code in main, all main does is call the function to take the users input.

Every function, except for ones that require user input, must be fully tested with one or more automated test functions. **You will be graded on the completeness of your automated tests.**

**Code guidelines:**

* **Good style:** Use proper indentation, good naming conventions and break code up into small functions that do one thing each.
* **Usability:** Always prompt the user for input so they know what to do and provide meaningful output messages.
* **Input Validation:** The program should not accept invalid input, prompt the user to reenter an input that is invalid.
* **Documentation:** Add a comments that document what each part of your code does, at a minimum each function should be clearly documented by describing the expected argument values and what the function returns.
* **Testing:** For this problem you must write automated tests and verify that they pass, see the problem statement.

Submit only the .cpp file containing the code.