In recent years the start of spring break has taken place 9 weeks after the start of the spring semester. Given a month, day, and year determine the date that occurs 9 weeks later.

Example Execution #1 (this school year):

Enter month number: 1
Enter day number: 11
Enter the year: 2016

Original Date: 1/11/2016
Future Date: 3/14/2016

Example Execution #2 (next school year):

Enter month number: 1
Enter day number: 9
Enter the year: 2017

Original Date: 1/9/2017
Future Date: 3/13/2017

Example Execution #3 (dates can span years):

Enter month number: 12
Enter day number: 31
Enter the year: 2015

Original Date: 12/31/2015
Future Date: 3/3/2016

Example Execution #4:

Enter month number: 10
Enter day number: 30
Enter the year: 2014

Original Date: 10/30/2014
Future Date: 1/1/2015

Additional Requirements:

1. **Add the homework assignment header file to the top of your program.** A description of your program will need to be included in the assignment header. This particular header can be added to your file by entering `hhw` while in command mode in `vi`.

2. The example executions provided each represent a single test of your program. We will test your final submission with the data used in the test case above and with several unpublished test cases.
   - Your program is expected to **accept input and produce output in the exact same manner demonstrated above**.
   - You may assume that the user of your program will only enter valid integers for the months, days, and years only in the range from 1800 – 2100.
   - Do not add any “bonus” features not demonstrated in the example executions provided.

3. Course standards **prohibit** the use of programming concepts not yet introduced in lecture. For this assignment you can consider all material in the **first five chapters** of the book, notes, and lectures to be acceptable for use.

4. For this assignment you will be **required** to implement the user-defined functions (from chapter 4). Failing to follow course standards as they relate to good user-defined function use will result in a **zero for this assignment**.

With our ability to implement user-defined functions **only the following** will be permitted in the main function:

1. Declaration of variables to be passed to functions.
2. Calls to user-defined functions by main.
3. A limited amount of control structures (see chapters 5 and 6) to retain the previous two tasks within the main function.

Additionally, each user-defined function may **represent a single task** in your larger program. The failure to make a good use of user-defined functions as described here and in the course standards will result in a loss of ALL points on those assignments that require user-defined functions.

5. A program **MUST** compile, be submitted through the guru server as demonstrated during the first week of the semester in lab, and submitted prior to the posted due date to be considered for partial credit.
Course Programming and Documentation Standards Reminders:

• Use the course function header (head_fx vi shortcut hfx while in command mode) for every user-defined function in your program.
  ◦ List and comment all parameters to a function, one per line, in the course function header.
  ◦ All function declarations will appear in the global declaration section of your program.
  ◦ The user-defined function definitions will appear in your program after the main function.

• Place a single space between all (binary) operators and operands.

• Comment all variables to the right of each declaration.

• Declare only one variable per line.

• Notice that several programs (see program 2-9 on pages 74-75) in the programming text use a single line comment to indicate the start of the local declaration and executable statement sections of the main function.
  ◦ At no point during the semester should these two sections ever overlap. You might consider adopting this habit of commenting the start of each section to help you avoid this mistake.

• Select meaningful identifiers (names) for all variables in your program.

• Indent all code found within the main function exactly two spaces.

• Indent all code found within the body of a selection construct exactly two additional spaces.

• Make use of { and } with all relevant section constructs.

• Do not single (or double) space the entire program, use blank lines when appropriate.

• There is no need to include example output with your submission.

• Remove any diagnostic print statements from your code, even if they are commented out (inactive), unless you believe your program to be logically incorrect or incomplete as a way to demonstrate to your lab instructor how much of the problem you were able to solve and attempted to implement.

When you submit... only the final attempt of a submission is kept for grading. All other submissions are over-written and cannot be recovered. You may make multiple submissions but only the last attempt is retained and graded.

• Verify in the confirmation e-mail sent to you by the course that you have submitted the correct file (must be named hw04.c), to the correct assignment (hw04), and to the correct lab section (CRN value).

• Leave time prior to the due date to seek assistance should you experience difficulties completing or submitting this assignment.

• All attempts to submit via a method other than through the guru server as set up during the first lab of the semester will be denied consideration.

Assignment deadlines... are firm and the electronic submission will disable promptly as advertised. We can only grade what you submit as expected prior to the assignment deadline.

Academic Integrity Reminder:

• Please review the policies of the course as they relate to academic integrity. The assignment you submit should be your own original work. You are to be consulting only course staff regarding your specific algorithm for assistance. Collaboration is not permitted on individual homework assignments.

All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document!