CS 159 – HW #04
Due: Monday March 15, 2021 at 11:00pm (time local to West Lafayette, IN).
10 Points Possible

Problem: Given as input the year, day of the week number (0 [Sunday] through 6 [Saturday]), and occurrence number, determine the corresponding date. On page 300 of your C programming text there are several useful formulas given as part of problem 60. All input will be integer data and your program will only be tested with years from 1800 to 2100.

Example Execution #1:

```
Enter year number -> 2021
Enter day of week number -> 1
Enter occurrence number -> 11

Desired occurrence #11 of Monday in 2021: March 15
```

Example Execution #2:

```
Enter year number -> 2020
Enter day of week number -> 6
Enter occurrence number -> 9

Desired occurrence #9 of Saturday in 2020: February 29
```

Example Execution #3:

```
Enter year number -> 2021
Enter day of week number -> 5
Enter occurrence number -> 1

Desired occurrence #1 of Friday in 2021: January 1
```

Example Execution #4:

```
Enter year number -> 2020
Enter day of week number -> 4
Enter occurrence number -> 53

Desired occurrence #53 of Thursday in 2020: December 31
```

Example Execution #5:

```
Enter year number -> 2020
Enter day of week number -> 3
Enter occurrence number -> 16

Desired occurrence #16 of Wednesday in 2020: April 15
```

Example Execution #6:

```
Enter year number -> 2019
Enter day of week number -> 0
Enter occurrence number -> 47

Desired occurrence #47 of Sunday in 2019: November 24
```

Example Execution #7:

```
Enter year number -> 2019
Enter day of week number -> 2
Enter occurrence number -> 40

Desired occurrence #40 of Tuesday in 2019: October 1
```

Example Execution #8:

```
Enter year number -> 2018
Enter day of week number -> 4
Enter occurrence number -> 22

Desired occurrence #22 of Thursday in 2018: May 31
```

Example Execution #9:

```
Enter year number -> 2018
Enter day of week number -> 5
Enter occurrence number -> 22

Desired occurrence #22 of Friday in 2018: June 1
```

Example Execution #10:

```
Enter year number -> 2017
Enter day of week number -> 2
Enter occurrence number -> 27

Desired occurrence #27 of Tuesday in 2017: July 4
```

Example Execution #11:

```
Enter year number -> 2016
Enter day of week number -> 3
Enter occurrence number -> 32

Desired occurrence #32 of Wednesday in 2016: August 10
```

Example Execution #12:

```
Enter year number -> 2000
Enter day of week number -> 6
Enter occurrence number -> 40

Desired occurrence #40 of Saturday in 2000: September 30
```

Example Execution #13:

```
Enter year number -> 1900
Enter day of week number -> 4
Enter occurrence number -> 9

Desired occurrence #9 of Thursday in 1900: March 1
```

Example Execution #14:

```
Enter year number -> 1900
Enter day of week number -> 3
Enter occurrence number -> 9

Desired occurrence #9 of Wednesday in 1900: February 28
```

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

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.

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. **Each of the example executions provided for your reference represents a single execution of the program.** Your program must accept input and produce output **exactly** as demonstrated in the example executions, do not add any "bonus" features not demonstrated in the example executions. Your program will be tested with the data seen in the example executions and an unknown number of additional tests making use of meaningful data.
3. 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.**
4. Revisit **course standards as it relates what makes for good use of user-defined functions, what is acceptable to retain in the** `main` **function, and when passing parameters by address is appropriate.**
   - In many cases user-defined function use should result in a `main` function that only declares variables and makes function calls.
5. 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.
6. A program **MUST** compile to be considered for partial credit. The submission script will reject the submission of any file that does not successfully compile on the `guru` server. The name of the source code file you attempt to submit must be `hw04.c`, no variation is permitted.

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.**
- Indent all code found within the `main` function **exactly** two spaces.
- Place a **single space** between all operators and operands.
- Comment **all** variables to the right of each declaration. Declare only one variable per line.
- In general it is acceptable to initialize a variable declared in the local declaration section of a function. If the expression used to initialize the variable is more complex than a constant assignment then it is best to give the variable its first value inside of the executable statement section of the function.
- 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.
- Select **meaningful identifiers** (names) for all variables in your program.
- Do not single (or double) space the entire program, **use blank lines when appropriate**.

When you submit... only the final successful 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 section.
- **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 in the Account Configuration Activity 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.

Auto-Grade Tool

- We have implemented what is being referred to as the auto-grade tool. At the time of a successful assignment submission you may receive some feedback on your program in regards to course programming and documentation standards. This feedback may include a potential deduction that you will receive once your assignment is reviewed by your grader.
- It is expected that graders verify those notes identified by this tool to ensure that they are indeed applicable and reasonable to the submission. Graders may make additional deductions for those standards not identified by the new tool.
- We hope that this feedback helps with the enforcement of course standards, consistency in grading across sections, and to encourage students to revise their work when problems are identified before the assignment deadline passes. It is possible to resubmit an assignment for grading up to the advertised deadline. Only the final successful submission is retained and evaluated.