CS 159 – HW #06
Due: Monday April 12, 2021 at 11:00pm (time local to West Lafayette, IN).
10 Points Possible

Problem: Given as input the value of ten years and a desired day of the week (0 – Sunday though 6 – Saturday) display the months within the first year that begin on the desired day of the week. For the second through the tenth year the desired day of the week will continually be incremented.

- 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 1700 to 2100. The day of the week number will be an integer and must be validated to be within the range from 0 to 6.

Example Execution #1:

```
Enter 10 years [1700 - 2100] -> 2021 2012 2020 2019 2013 2014 2018 2015 2017 2016
Enter day of week number -> 1

Desired day of the week: Monday
Months in 2021 starting on Monday: February March November
Months in 2012 starting on Tuesday: May
Months in 2020 starting on Wednesday: January April July
Months in 2019 starting on Thursday: August
Months in 2013 starting on Friday: February March November
Months in 2014 starting on Saturday: February March November
Months in 2018 starting on Sunday: April July
Months in 2015 starting on Monday: June
Months in 2017 starting on Tuesday: August
Months in 2016 starting on Wednesday: June
```

Example Execution #2 (input validation expectations demonstrated):

```
Enter 10 years [1700 - 2100] -> 2000 1996 1992 1988 1984 2004 2008 2012 2016 2020
Enter day of week number -> 9

Error! Day of the week must be 0 to 6.

Enter day of week number -> -1

Error! Day of the week must be 0 to 6.

Enter day of week number -> 4

Desired day of the week: Thursday
Months in 2000 starting on Thursday: June
Months in 1996 starting on Friday: March November
Months in 1992 starting on Saturday: February August
Months in 1988 starting on Sunday: May
Months in 1984 starting on Monday: October
Months in 2004 starting on Tuesday: June
Months in 2008 starting on Wednesday: October
Months in 2012 starting on Thursday: March November
Months in 2016 starting on Friday: January April July
Months in 2020 starting on Saturday: February August
```

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.
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 may be tested with the data seen in the example executions and an unknown number of additional tests making use of meaningful data as specified in the problem description.
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.** Good function design will limit each function to a single task, eliminate redundant logic in the program, and maximize re-use of functions within a program.
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 calls functions. Selection and repetition constructs in a `main` function are to facilitate the calling of functions.
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 eight chapters** of the book, notes, and lectures to be acceptable for use.
   - The use of **character arrays**, including character pointers (chapter 11) to represent string data, would violate requirements of this assignment and **result in no credit being awarded for your effort.**
   - **The use of** any dynamic array structures (chapters 9 and 10) would violate this requirement and result in **no credit being awarded for your effort.** See course standards below for array declaration expectations.
   - **Only a single integer array of size 10 is needed to solve this problem.**
6. A program **MUST** compile, be submitted through the `guru` server prior to the posted due date 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 `hw05.c`, no variation is permitted.

Selected Course Programming and Documentation Standards Reminders:

- It is common to make use of a symbolic/defined constant when the size of the array is known prior to the start of a program.
- The course standards expect all arrays to be of a fixed size. Variable-size arrays, even those demonstrated in chapter 8 of the text, would violate course standards.
- Code found inside the body of relevant selection and repetition constructs must be indented two additional spaces.
- Make use of `{` and `}` with all relevant selection and repetition constructs.
- See page 258 of your C programming text regarding the proper indentation for a `switch` construct.
- 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.**
- Maximize your use of symbolic/defined constants and minimize your use of literal constants.
- 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.
- At no point during the semester should the local declaration and executable statement sections of a function ever overlap.
- Select **meaningful identifiers** (names) for all variables in your program.

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 `hw06.c`), to the correct assignment (`hw06`), 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.