Problem Set #2

Assignment Learning Objectives

1. Write a program that uses selective and repetitive control to solve higher complexity programming problems and reads input from a file
2. Apply the correct programming tools to efficiently solve problems
3. Practice robust computer coding strategies

Due 8 September 2020 by 3:00 pm

Due Online

- One script in the PSet2 repository on your GitHub account

Due Via Blackboard

- A PDF version of your code. Make sure the honor code appears in the comment block at the beginning of the script and that it has your full typed out name. This will serve as your assertion that you have upheld the honor code.

Honor Code

You must type the full honor code into each comment block to indicate that you have upheld the code. Authorized aid for this assignment is your notes and any previous programs you have written. You are highly encouraged to work with others in the class to help diagnose bugs and work out programming logic. Any copying of someone else's code IS A VIOLATION OF THE HONOR CODE, whether from a classmate, or the Internet. *Be sure to indicate with whom you have worked in the comment block of your submission.*

Background

Progress in science is based on the collecting and analyzing of data and meteorology is a clear example of a data-driven science. With meteorological data, *time* plays an inordinately large role in properly describing the atmospheric conditions due to the large daily, weekly, monthly, yearly, etc. variability. Most elements of time area consistent and not terribly difficult to work with; however, with long period meteorological records a notorious problem lurks within the data, *leap day*. In the vast majority of years there are only 365 days, but Earth actually travels around the sun in 365 days, 5 hours, 48 min, and 46 sec. In order to account for the extra time we add a day (leap day) in some years, according to a set of rules, to make up for lost time.

Leap Year Definition

Add a leap day every four years when the year is divisible by 4, but not if the year is divisible by 100 unless it is also divisible by 400.

Due to the vastly fewer numbers of leap days (Feb 29) within meteorological records, some datasets don't include it! Yet, it is an actual day that must be accounted for in many circumstances.

Another aspect of time that needs special attention is the day of the year. The common way to identify a day is by using both the month and day (e.g., 6 June), but that is relatively complicated when really it is just one day in a sequence of 365 (or 366 in a leap year). Therefore, there are certain segments of the meteorological community (e.g., satellite meteorology) that use the *Julian Day* of the year. Note: This is very different from Julian Date, which is the continuous number of days since 1 January 4713 BC on the Julian calendar.

Julian Day Definition

> The day of the year as a single numeric value between 1 and 365 in non-leap years, starting on 1 January, increasing by 1 each day through 31 December.
>
> https://landweb.modaps.eosdis.nasa.gov/browse/calendar.html

Problem

1. Write a program that will determine if a series of ten (10) daily high temperatures are normal (between the first and third quartile, inclusive), well above/below average (above [below] the third [first] quartile, but not record level), or record breaking for the meteorological season in which they occur.

   Program Name: **climate_check.for**

   Program Input: From a file read a series of year, month, day, high temperature in Fahrenheit

   Program Output:

- Date in Year, Month, Day format
- Leap Year Indicator
- Julian Day
- Indication of "Normal Temperature", "Well Above Average Temperature", "Well Below Average Temperature", "Record Temperature" with the meteorological season that it occurred in.

> Meteorological Seasons
>
> Winter: December, January, February
>
> Spring: March, April, May
>
> Summer: June, July, August
>
> Fall: September, October, November
>
> Climate Statistics for VPZ (°F)

Season	**Record High**	**75%**	**50%**	**25%**	**Record Low**
Winter	71	41	34.2	28	-10
Spring	96	71	58.6	47	13
Summer	105	87	81.7	77	52
Fall	101	74	62.8	52	10

Notes:

- Break the problem down into smaller bite size pieces. For example, first write your program to be able to determine if a year is a leap year from a command line input. Then submit that working code to your GitHub repository for this assignment. Now go back and read a series of years from a file, and so on.
- Fortran has a number of built in functions, some of which might be useful in this case. Specifically a function called "MOD" and "SUM" might be of interest.
- Make sure to regularly submit your work to your GitHub site. Part of your grade on each assignment is on your regular submission of the program(s) as they are in progress of being developed. This is easily done when regularly submitting the bite size pieces that you create when breaking down any coding project!
- Test more than just the cases or example file given below. An important part of code development is testing to make sure your code works for ALL cases, so think about "edge" cases that you know to be problematic and test those specifically.

  Please make your program to produce output in the same fashion as the test cases below.

  Test Case 1

  > Begin Record <

  Year Month Day

  Date: 2000 2 18

  2000 is a leap year

  Julian Day of the year: 49

  Temperature (F): 72.000000

  Climate records indicate that this temperature is

  a record warm temperature for winter.

  > End Record <

  Test Case 2

  > Begin Record <

  Year Month Day

  Date: 1995 7 4

  1995 is a leap year

  Julian Day of the year: 185

  Temperature (F): 85.000000

  Climate records indicate that this temperature is

  within normal range for summer.

> End Record <

Input Examples:

---

year, month, day, high temperature (F)

1900, 9, 28, 45

2017, 6, 19, 105

1972, 1, 13, -10

1988, 4, 20, 55

2003, 8, 17, 45

2004, 3, 24, 68

1955, 11, 2, 28

1968, 12, 14, -5

Evaluation Criteria

Each of the following criteria will be rated from not present/completed to exemplary, having all of the elements will yield at least a 7/10 for a particular criterion. The assignment is out of 50 points.

- Leap year is correctly determined, coded efficiently, styled well, and reported in output (10 points)
- Efficiently coded, styled well, and correct reporting of the input temperature from the file (10 points)
- Efficiently coded, styled well, and correct interpretation of the temperature's seasonal normality (10 points)
- Code is tested and complies/runs without error (10 points)
- Code is well documented (informative comments/descriptions of code blocks; 5 points)
- Code was regularly updated and committed to GitHub repository (5 points)