# **CS211 Fall 2022 Lab-7: Structures**

# **Lab-7 Setup**

**Create "lab-7" directory on U-Drive**

* Open a new terminal window to SSH into your account on the [bingweb.binghamton.com] server. When logged in, make sure you are in your home directory by typing the following command if you are not already there. The "~" is just a shortcut symbol that means home directory:  
  cd ~
* Create a directory named "lab-7" by typing:  
  mkdir lab-7
* Change your current directory to the "lab-7" directory by typing:  
  cd lab-7
* Confirm that you are in the "lab-7" directory by typing:  
  pwd
* Download the attached “lab7.c” and “README.md” files and move them to the “lab-7” directory on your U-Drive.

**Lab-7 Content**

Lab-7 consists of 3 files. One is provided to you, and the other two you should implement. The files are:

* lab7.c (provided)
* lab7structures.h (should be implemented)
* lab7structures.c (should be implemented)

**Lab-7 Implementation Guidelines**

You should write your code in “lab7structures.c” and “lab7structures.h”. ***Do not make any changes to the source code in the "lab7.c" file.*** If you do so, your program will not run successfully, and you will lose points.

To compile, make sure all files are in the "lab-7" directory, then run the command:  
gcc lab7.c lab7structures.c -o lab7.out

Because header files are included in the source “.c” files, they are not included in the "gcc" command to get compiled.

To run lab-7, type the command:  
./lab7.out

# **Implementation**

**Header File**

Add a header file to your project directory “lab-7”. To do this, follow the following steps:

* Create a new header file in your project directory called "lab7structures.h".
* Add include guards to the header file.
* Copy and paste the following to your header file:  
  #include <stdio.h>  
  #include <assert.h>  
  #include <string.h>  
  #define STR\_SIZE 20  
  typedef struct  
  {  
   unsigned int year;  
   unsigned int month;  
   unsigned int day;  
  } Date;

**Source Code File**

“lab7.c” calls two functions that depend on four additional functions. You should implement all six functions in “lab7structures.c” and declare them in “lab7structures.h”. Each of these functions is described below. Now, begin by adding a new source code file to your project directory “lab-7”. To do this, follow the following steps:

* Create a new source code file in your project directory called "lab7structures.c".
* Include the header file “lab7structures.h” in the source code file “lab7structures.c”.
* Implement the following functions in “lab7structures.c”.

In Lab-7, we will implement a program that calculates:

* The number of elapsed days between two dates.
* On which day of the week a particular date falls.

In the header file, we define a new data type called “Date”. “Date” is a structure that consists of three members; “year”, “month”, and “day”. All members are unsigned integers.

**unsigned int validate(Date d)**

Implement the function "*unsigned int validate(Date d)*" to check if the given date in the structure “d” is valid. The function should return “1” if the date is valid or return “0” if the date is not valid.

The function should do the following:

* **Return “0”:** if any of the following conditions hold
  + The “year” is either less than 1901 or greater than 2020.
  + The “month” is equal to 0 or greater than 12.
  + The “day” is equal to 0.
  + The “month” is February and the “day” is greater than 29.
  + The “month” is April, June, September, or November, and the “day” is greater than 30.
  + The “month” is January, March, May, July, August, October, or December, and the “day” is greater than 31.
* **Return “1”:** if none of the above conditions are true.

**unsigned int calculateYear(Date d)**

Implement the function "*unsigned int calculateYear(Date d)*" to calculate a value based on the year of the given date in the structure “d”. The function should return the computed value, which is of type unsigned integer.

The function should do the following:

* **Return “year - 1”:** if the “month” is January or February.
* **Return “year”:** if the “month” is ***not*** January or February.

**unsigned int calculateMonth(Date d)**

Implement the function "*unsigned int calculateMonth(Date d)*" to calculate a value based on the month of the given date in the structure “d”. The function should return the computed value, which is of type unsigned integer.

The function should do the following:

* **Return “month + 13”:** if the “month” is January or February.
* **Return “month + 1”:** if the “month” is ***not*** January or February.

**unsigned int calculateN(Date d)**

Implement the function "*unsigned int calculateN(Date d)*" to calculate the “N” value based on the formula below for the given date in the structure “d”. The function should return the computed “N” value, which is of type unsigned integer.

The function should do the following:

* **Calculate “N”:** use the following formula to calculate the “N” value. The formula depends on the return value of “calculateYear()” function, the return value of “calculateMonth()” function, and the “day” value for the given date in the structure “d”
* **Return “N”.**

**unsigned int elapsedDays(Date d1, Date d2)**

Implement the function "*unsigned int elapsedDays(Date d1, Date d2)*" to calculate the number of elapsed days between two given dates in the structures “d1” and “d2”. The function should return the computed number of days, which is of type unsigned integer.

The function should do the following:

* **Check Dates:** use the function “validate()” to check whether the two given dates, “d1” and “d2”, are valid or not. If any of the dates is not valid, return 0.
* **Calculate “N”:** use the function “calculateN()” to compute the “N” value for the two given dates, “d1” and “d2”.
* **Calculate the Difference:** calculate the difference between the “N” values of the two given dates, “d1” and “d2”.
* **Return the Difference.**

Your function will be tested using the following values:

* d1 = 2/1/1900, d2 = 8/5/2004 ⇒ return = 0.
* d1 = 8/8/1988, d2 = 1/1/2021 ⇒ return = 0.
* d1 = 13/8/1950, d2 = 10/15/2004 ⇒ return = 0.
* d1 = 2/10/1920, d2 = 1/0/1970 ⇒ return = 0.
* d1 = 7/1/2010, d2 = 7/1/2010 ⇒ return = 0.
* d1 = 12/31/1999, d2 = 1/1/2000 ⇒ return = 1.
* d1 = 7/2/2015, d2 = 7/16/2015 ⇒ return = 14.
* d1 = 8/8/2004, d2 = 2/26/2005 ⇒ return = 202.
* d1 = 11/30/1958, d2 = 1/31/1959 ⇒ return = 62.
* d1 = 1/1/1901, d2 = 12/31/2020 ⇒ return = 43829.

**unsigned int dayOfWeek(Date d, char daystr[])**

Implement the function "*unsigned int dayOfWeek(Date d, char daystr[])*" to calculate on which day of the week a particular date falls. The function receives as parameters a given date in the structure “d” and an array of characters “daystr”, which will hold the day of the week after it is computed. The function should return “1” if the computation was successful or return “0” if the given date is not valid.

The function should do the following:

* **Check Date:** use the function “validate()” to check whether the given date, “d”, is valid or not. If the date is not valid, return 0.
* **Calculate “N”:** use the function “calculateN()” to compute the “N” value for the given date, “d”.
* **Calculate Day Value:** calculate the day value by subtracting “621049” from the “N” value.
* **Determine Day of Week:** divide the day value by “7” to get a remainder between 0-6, which corresponds to Sunday-Saturday. Save the corresponding day of week string in “daystr” array, then return 1.

Your function will be tested using the following values:

* d = 12/31/1900 ⇒ daystr = “”, return = 0.
* d = 3/1/1905 ⇒ daystr = “Wednesday”, return = 1.
* d = 5/12/1985 ⇒ daystr = “Sunday”, return = 1.
* d = 7/9/2004 ⇒ daystr = “Friday”, return = 1.
* d = 9/20/2018 ⇒ daystr = “Thursday”, return = 1.
* d = 11/10/2018 ⇒ daystr = “Saturday”, return = 1.
* d = 12/9/2019 ⇒ daystr = “Monday”, return = 1.
* d = 12/22/2020 ⇒ daystr = “Tuesday”, return = 1.

Save your program, compile and run it, and fix errors if you get any.

**Valgrind**

Run your code with Valgrind on the Linux server, and ensure you have no memory errors.

To run your code with Valgrind, make sure all files are in the "lab-7" directory, then compile your code using this command **Notice the extra flag, -Wall, this will list all warnings**:  
gcc -g lab7.c lab7structures.c -Wall -o lab7.out

Now run lab-7 with Valgrind using this command:  
valgrind --leak-check=full ./lab7.out

If you have any memory errors, correct them before submitting.

# **EXTRA CREDIT (20%):**

Write the following functions:

Function 1: Write a function (void return), named initializeArray that will take two parameters. The first parameter will be an integer array and the second parameter will be the size of the array.

The function will assign a random number between 0 and 500 to each element of the array.

Function 2: Write a function (void return), named printTable that will take two parameters. The first parameter will be an integer array (should be const) and the second parameter will be the size of the array.

The function will print the array as a 10x10 table (Think back to one of your discussions). Each row will need to labeled starting (row 1…row10) and each column needs to be labeled (col 1…col 10). The row number should use a variable. (I’ll let you figure out how to do it)

Function 3 and 4: Implement the selection sort that was used in class. The example and notes should be on Brightspace. You MUST use the selection sort provided, using any other sort will result in 0 extra credit points.

In your header File: Declare a constant or macro for the size of the array and set this to 100. Declare each of your prototypes.

In your main function:

At the top of the main, declare the integer array using your constant/macro.

At the bottom of your main, call Function 1, call Function 2, call Function 3 (this will call Function 4) and then call Function 2 again.

Your output should show two tables. One table will be of the array before it is sorted and one will be of the array after it is sorted.

# **Submission**

* Complete the "README.md". You may write ‘n/a’ where applicable (bugs, references, etc.). There is no need to keep the "< >" symbols---they’re just there to signify that you should replace the text there.
* Please only submit the following files for Lab-7 on Brightspace using the same file name and extension as here:
  + lab7.c
  + lab7structures.h
  + lab7structures.c
  + README.md
* You can update your submission as often as you like until the deadline. I strongly recommend submitting early on to avoid late penalties, even if your assignment is not 100% working. You must submit your lab before the deadline to be considered on time. Otherwise, it will be considered late even if your lab was completely working before the deadline. ***Please note that on Brightspace, we maintain all your submissions but only grade your most recent submission.***
* That's it! We've completed our work for this lab.

# **Grading Rubric**

Total: 40 points

* Header file “lab7structures.h”: 2 points (1 point if partially completed)
* Implement validate() function: 4 points
* Implement calculateYear() function: 2 points
* Implement calculateMonth() function: 2 points
* Implement calculateN() function: 4 points
* Pass elapsedDays() tests: 8 points
* Pass dayOfWeek() tests: 9 points
* Pass Valgrind with no memory errors: 5 points
* Submission: 2 points (1 point if partially completed)
* Compiling with -Wall and receiving 0 warnings: 2 points

EXTRA CREDIT: 8 points

Any extra credit that does not produce correct output and/or does not use the selection sort will receive 0 points.

Other Infractions:

Not documenting the code (-2)

Not providing a label for the rows: (-1)

Not providing a label for the columns (-1)

Not using the constant/macro declared in the header file (-1)

(This should be used in the declaration and when calling the function)