General Notes

- You will submit a minimum of three files, the core files must conform to the following naming conventions (including capitalization and underscores). 123456789 is a placeholder, please replace these nine digits with your nine-digit Bruin ID. The files you must submit are:

  1. *123456789_stats102a_hw1.R*: An R script file containing all of the functions you wrote for the homework. The first line of your .Rmd file, after loading libraries, should be sourcing this script file.
  2. *123456789_stats102a_hw1.Rmd*: Your markdown file which generates the output file of your submission.
  3. *123456789_stats102a_hw1.html/pdf*: Your output file, either a PDF or an HTML file depending on the output you choose to generate.
  4. *included image files*: You may name these what you choose, but you must include all of the image files you generated for your structured flowcharts, otherwise your file will not knit.

  If you fail to submit any of the required core files you will receive ZERO points for the assignment. If you submit any files which do not conform to the specified naming convention, you will receive (at most) half credit for the assignment.

- Your .Rmd file must knit. If your .Rmd file does not knit you will receive (at most) half credit for the assignment.
  The two most common reason files fail to knit are because of workspace/directory structure issues and because of missing include files. To remedy the first, ensure all of the file paths in your document are relative paths pointing at the current working directory. To remedy the second, simply make sure you upload any and all files you source or include in your .Rmd file.

- Your coding should adhere to the tidyverse style guide: https://style.tidyverse.org/.

- All flowcharts should be done on separate sheets of paper, but be included, inline as images, in your final markdown document.

- Any functions/classes you write should have the corresponding comments as the following format.

```
my_function <- function(x, y, ...){
#A short description of the function
#Args:
#x: Variable type and dimension
#y: Variable type and dimension
#Return:
#Variable type and dimension
Your codes begin here
}
```

NOTE: *Everything* you need to do this assignment is here, in your class notes, or was covered in discussion or lecture.

- **DO NOT** look for solutions online.
- **DO NOT** collaborate with anyone inside (or outside) of this class.
- Work **INDEPENDENTLY** on this assignment.
- **EVERYTHING** you submit **MUST** be 100% your, original, work product. Any student suspected of plagiarizing, in whole or in part, any portion of this assignment, will be **immediately** referred to the Dean of Student's office without warning.

Homework Requirements

a. Create a structured flowchart of the algorithm.

b. Write pseudocode sufficiently complete, clear, and concise enough to enable a person to accurately implement the algorithm in any programming language they are adept with using.

c. Write the function(s) which accurately implements the algorithm(s) as described or requested.

d. Include the error-handling to ensure your function(s) work properly.

1: Greatest Common Divisor (GCD)

Please write a function, `gcd()`, which takes two positive integers a, b and calculates their greatest common divisor (GCD) using the Euclidean algorithm.

The Euclidean Algorithm Example:
Let a = 180 and b = 25

1. calculate 180/25, and get the result 7 with remainder 30, so $180 = 7 \times 25 + 5$.
2. calculate 25/5, and get the result 5 with remainder 15, so $25 = 5 \times 5 + 0$.
3. the greatest common divisor of 180 and 25 is 5.

Based on your `gcd()`, please write another function, which takes three positive integers and returns their GCD.

2: Prime Factorization

Please write a function `get_factors()` which takes a positive integer x and returns the vector of (not necessarily unique) prime factors of x. In addition, write at least one "helper function" `is_prime()` which returns TRUE or FALSE depending on whether or not the number x is prime.

A good way to test this function should be:

```
x <- sample(x = 1e4, size = 1)
y <- get_factors(x)
this_works <- prod(y) == x & all(is_prime(y))
```

It is a necessary, but not sufficient, condition that `this_works == TRUE` for your function to work as intended. (It will largely depend on if your `is_prime()` function is correct.)