Instructions

Part 1
For this part of the assignment you are asked to replicate two basic **R** functions – one is the `sum()` function and the other the `cumsum()` function. As you know the `sum()` function adds all the elements of a vector together. The `cumsum()` function is slightly more complicated. Rather than returning a vector of length 1 containing the sum, the `cumsum()` function returns a vector of the same length as the input vector. Each element of the output vector contains the sum of an element of the input vector and all elements that appear before it in the vector.

Consider the following example:

```
> A <- 1:10
> A
[1]  1  2  3  4  5  6  7  8  9 10

> B <- cumsum(x = A)
> B
[1]  1  3  6 10 15 21 28 36 45 55
```

The purpose of this task is to review both functions and for-loops, though you might think of solutions that do not require for-loops.

A) Adding elements of vectors

Produce a function that replicates the `sum()` function.[1]

```
> x <- c(25, 2, 19, 81)
> y <- c(pi, -99, sqrt(x = 1:10))
> my_sum <- ...


> my_sum(input = x) == sum(x = x)
[1] TRUE
> my_sum(input = y) == sum(x = y)
[1] TRUE
```

B) Adding elements of vectors – slightly more complex

Produce a function that replicates the `cumsum()` function.[2]

```
> x <- c(25, 2, 19, 81)
> y <- c(pi, -99, sqrt(x = 1:10))
> my_cumsum <- ...
```

[1] The `sum()` function simply adds all the elements of a numeric vector and returns as its output the sum as a numeric vector of length 1.

[2] The `cumsum()` function returns a vector of the same length as the input vector containing at location i the sum of all elements of the input vector stored before i and at i. In other words, the `cumsum()` function produces a vector of cumulative sums. I am sure this description does not make much sense so you may want to try the `cumsum()` function with a few vectors to make sure you understand what it is doing.

```
> my_cumsum(input = x) == cumsum(x = x)
[1] TRUE
> my_cumsum(input = y) == cumsum(x = y)
[1] TRUE
```

Part 2

In class we wrote a function that simulates the rolling of dice and then used that function repeatedly (via a repeat-loop) trying to determine how many times we have to roll three dice until we got three 2s. We noticed that depending on random chance, we had to roll three dice quite a few times to come up with three 2s. Moreover, it sometimes required only a handful of trials or rolls, and at other times hundreds or even thousands of trials until we got lucky and our repeat-loop "broke" or stopped.

For Part 2, I want you to recreate this repeat-loop and determine how many tries or rolls it takes to get a 1, a 2, and a 3 in any order. The repeat-loop should break the moment we roll either (1, 2, 3), (3, 2, 1), (2, 1, 3), (3, 1, 2), (2, 3, 1) or, (1, 3, 2).[3]

Once you have the above worked out, I want you to run this repeat-loop 1,000 times. Each time, record the number of trials it took to get a 1, a 2, and 3 in any order and store this information in a vector. In other words, create a vector of length 1000, containing 1000 counts of trials. Compute the average and standard deviation for this vector. In other words, tell me how long it typically takes to roll a 1, a 2, and 3 in any order, and how much we can expect the number of attempts it takes to roll a 1, a 2, and 3 to vary.

Extra-Credit

For extra-credit. Determine how many dollars I have to spend to win the grand prize of the Powerball lottery once with the following combination of numbers and a random seed of 666.

- Numbers: 10, 20, 30, 40, 50
- Powerball: 13

Assume a lottery ticket costs $2. Also, as a bit of information, to win the grand prize of the Powerball lottery your five numbers and the powerball have to match the numbers and powerball that are drawn. The five numbers are drawn without replacement from 1 to 69 and the powerball is a separate random draw from 1 to 26. The order of the five numbers doesn't matter. Prior to running your program to find the dollar amount, set the random number seed to 666 and make sure your computer is fully charged.

[3] Hint: Don't overthink this, this does not require a nested if-else construct. Recall the the `is.element()` and `all()` and `any()` functions.