Problem 5

Yufei Yin

Problem 5

Control flow and functions are a basic aspect of every computer language. In R you can use loops with the following syntax: for (A in B) { C }. Here, if 'A' is a variable name, and 'B' is a vector and 'C' is an expression, then the expression will be evaluated once for each element of the vector 'B' (and the variable 'A' may be used in the expression 'C' and the value of 'A' will walk through all elements of the vector 'B'). You can define and call functions with the syntax 'A = function(B) { C; return(D); }'. In this code, 'A' is now a function that takes one argument, and 'C' and 'D' are expressions that may refer to 'B'. You can call the function 'A' using code like 'A(7)'. Here's an example:

```
A = function(B) { C = B + 7; return(C)}
A(22)
```

```
## [1] 29
```

The result of 'A(22)' should be 29, does that make sense?

Question 5a, (3 points):

Fibonacci numbers. The first two Fibonacci numbers are both 1 ($f_1 = f_2 = 1$). The n-th Fibonnaci number f_n (for n > 2) is $f_n = f_{n-1} + f_{n-2}$.

Write R code to compute the sum of the first 20 Fibonacci numbers. You may use recursive functions, or you may use Cramer's rule for linear equations.

Provide the sum of the first 20 Fibonacci numbers.

Question 5b, (1 point):

Provide the R code you used to compute the previous answer.

Question 5c, (1 bonus point):

Provide an estimate of the logarithm of the sum of the first one million Fibonacci numbers, in scientific notation with 4 digits in the significand.

```
Fibonacci <- function(n) {
  if ((n == 1)|(n == 2)) {return(1)}
  else { return(Fibonacci(n-1) + Fibonacci(n-2)) }
}

x <- NULL
for (i in 1:20) {
  x[i] <- Fibonacci(i)
}
sum(x)
```

```
## [1] 17710
```

```
# y <- NULL
# for (i in 1:1e6) {
#   y[i] <- Fibonacci(i)
# }
# sum(y)
# signif(log(sum(y)), digits = 4)
```