```
library(RSQLite)


# Q6a

# Slow: Define fib(n) to find n-th Fibonacci number
fib = function(n){ # use recursion
  if (n <= 1) {return(n)} # fib(0) = 0, fib(1) = 1
  else {return (fib(n-1) + fib(n-2))}
}

# Define sum_fib(n) to find the sum of first n Fibonacci numbers
sum_fib = function(n){
  sum = 0 # Start an int to store fib numbers
  for (i in seq(1,n)){
    sum = sum + fib(i)
  }
  return(sum)
}

sum_fib(20) # the sum of the first 20 Fibonacci numbers is 17710


# The following two methods could be used in further implications that
# might requires big numbers as recursion is horribly slow.

# Fast: Matrix multiplication
# [1 1]^n
# [1 0]
# will produce:
# [F(n+1) F(n)  ]
# [F(n)   F(n-1)]

# library(expm)
# fibm <- matrix(c(1,1,1,0), ncol=2, nrow=2)
# fibm %^% 1000

# Fast: Binet's
# fib_2 = function(n){
#  return((((1+sqrt(5))/2)^n-((1-sqrt(5))/2)^n)/sqrt(5))
# }
# signif(fib_2(1000),4) # 4.3467 E+208



################################################################




# Q6c:

# Many tries:
```

```r
# log(sum_fib(1000000)) # System.Out.Time
# log(sum_fib(20)) # 9.781885
# fib(1000) # System.Out.Time







# Faster: Using a while loop without recursion
# To deal with extreme numbers, use as.bigz() in "gmp" package
library(tidyverse)
library(gmp)


x = as.bigz(0)
y = as.bigz(1)
v = c(x,y)
i = 1
sum = as.bigz(0)

while (i <= 1000000){
  sum = sum+y
  i = i + 1
  v = c(y,x+y)
  x = v[1]
  y = v[2]
}

# sum # first 10 digits is 5113759002.....
# nchar(as.character(sum)) # 208988 digits in the sum

log.bigz(sum) # log of the sum gives an estimate of 481212 with no decimal
placed
# this means e^481212 ≈ sum
# An estimate in the scientific notation with 4 significant figures:
# 4.812 * 10^5
```