Homework3

1. Suppose we have a dataset A (see code below) where each column represents multiple measures of nitrogen concentration in a particular lake. We want to get the average value for each lake. Do this in two ways: a for loop and a vectorized function colMeans().

```
set.seed(12) # to be reproducible
A = matrix(data = runif(n = 1:500), nrow = 50, ncol = 10)
colnames(A) = paste("lake", 1:10, sep = "_")

##First using a loop##
for(i in 1:ncol(A)){
  lakemeans <-(mean(A[ , i]))
}
# paste lake name and mean
 print(paste(colnames(A), lakemeans, sep = " "))
```

```
##  [1] "lake_1 0.485641260328703"  "lake_2 0.485641260328703"
##  [3] "lake_3 0.485641260328703"  "lake_4 0.485641260328703"
##  [5] "lake_5 0.485641260328703"  "lake_6 0.485641260328703"
##  [7] "lake_7 0.485641260328703"  "lake_8 0.485641260328703"
##  [9] "lake_9 0.485641260328703"  "lake_10 0.485641260328703"
```

```
##Now using colMeans##
colMeans(A)
```

```
##    lake_1    lake_2    lake_3    lake_4    lake_5    lake_6    lake_7    lake_8
## 0.4601492 0.4992815 0.5987037 0.4580486 0.4719578 0.4965216 0.5110536 0.4577936
##    lake_9   lake_10
## 0.5193423 0.4856413
```

2. From the for loop lecture, we see the following example of using apply():

```
x = array(1:27, dim = c(3, 3, 3))
apply(X = x, MARGIN = c(1, 2),
      FUN = paste, collapse = ", ")
```

```
##      [,1]         [,2]         [,3]
## [1,] "1, 10, 19" "4, 13, 22" "7, 16, 25"
## [2,] "2, 11, 20" "5, 14, 23" "8, 17, 26"
## [3,] "3, 12, 21" "6, 15, 24" "9, 18, 27"
```

Now, use for loops to get the same task done (hint: nested loops). The results should be the same.

```
x = array(1:27, dim = c(3, 3, 3))
#Create empty array to fill
x.fill = array(dim = c(3, 3, 1))
#for loop to pull the number from each location in the original array, then paste the numbers from the
for (i in 1:3){
  for (j in 1:3){
    x.fill[i, j, ] = paste(x[i, j, 1], x[i, j, 2], x[i, j, 3], sep = ",")
    final.array <- x.fill
    }
  }

print(final.array)
```

```
## , , 1
##
##      [,1]       [,2]       [,3]
## [1,] "1,10,19" "4,13,22" "7,16,25"
## [2,] "2,11,20" "5,14,23" "8,17,26"
## [3,] "3,12,21" "6,15,24" "9,18,27"
```

3. The Fibonacci Sequence is the series of numbers that the next number is the sum of the previous two numbers: 0, 1, 1, 2, 3, 5, 8 . . . Use a for loop to get the first 30 numbers of the Fibonacci Sequence. This question should demonstrate the need for loops because there is no easy way to use vectorized functions in this case.

```
#create a numeric variable
fibonacci <- numeric(30)
#create a for loop to add the previous two numbers together
fibonacci[1] <- fibonacci[2] <- 1
for (i in 3:30){
  fibonacci[i] <- fibonacci[i-1]+ fibonacci[i-2]
}
print(fibonacci)
```

```
##  [1]      1      1      2      3      5      8     13     21     34     55
## [11]     89    144    233    377    610    987   1597   2584   4181   6765
## [21]  10946  17711  28657  46368  75025 121393 196418 317811 514229 832040
```

4. In the example data below, extract those ranking numbers with regular expression. The results should have the number(s) and . if it follows after the numbers immediately (i.e., 1., 12., 105., 105.3, etc.). Remove empty strings from the final results. You should get 107 strings for your results.

```
top105 = readLines("http://www.textfiles.com/music/ktop100.txt")
top105 = top105[-c(64, 65)] # missing No. 54 and 55
##use expression from class ##
N <- "^\\d\\.?\\d?\\d?\\.?"
combine <- regexpr(N, top105)
List <- regmatches(top105, combine)
print(List)
```

```
##   [1] "1."   "2."   "3."   "4."   "5."   "6."   "7."   "8."   "9."   "10."
```

```
##  [11] "11."  "12."  "13."  "14."  "15."  "16."  "17."  "18."  "19."  "20."
##  [21] "21."  "22."  "23."  "24."  "25."  "26."  "27."  "28."  "29."  "30."
##  [31] "31."  "32."  "33."  "34."  "35."  "36."  "37."  "38."  "39."  "40."
##  [41] "41."  "42."  "43."  "44."  "45."  "46."  "47."  "48."  "49."  "50."
##  [51] "51."  "52."  "53."  "56."  "57."  "58."  "59."  "60."  "61."  "62."
##  [61] "63."  "64."  "65."  "66."  "67."  "68."  "69."  "70."  "71."  "72."
##  [71] "73."  "74."  "75."  "76."  "77."  "78."  "79."  "80."  "81."  "82."
##  [81] "83."  "83."  "84."  "85."  "86."  "87."  "88."  "89."  "90."  "91."
##  [91] "91."  "92."  "93."  "94."  "95."  "96."  "97."  "97."  "98."  "99."
## [101] "100." "101." "102." "103." "104." "105." "105."
```

5. For the vector with length of 107 you got from question 4, remove all trailing .. (hint: ?sub). Then convert it to a numeric vector and find out which numbers have duplications (i.e., a tie in ranking). Don't count by eyes, use R to find it out (hint: table(), sort(); or duplicated(), which(), [subsetting; there are more than one way to do so).

```
### Remove "."
sub(pattern = "\\.", replacement = "", x = List)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"
##  [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"
##  [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
##  [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"
##  [49] "49"  "50"  "51"  "52"  "53"  "56"  "57"  "58"  "59"  "60"  "61"  "62"
##  [61] "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"
##  [73] "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "83"  "84"  "85"
##  [85] "86"  "87"  "88"  "89"  "90"  "91"  "91"  "92"  "93"  "94"  "95"  "96"
##  [97] "97"  "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "105"
```

```
### Make numeric
NumList <- as.numeric(List)
### Print duplicated numbers
print('Duplicated numbers:')
```

```
## [1] "Duplicated numbers:"
```

```
NumList[duplicated(NumList)]
```

```
## [1]  83  91  97 105
```