MACHINE LEARNING

NAME: ADITYA RANJAN

REG. NO. : 16BCE2310

DIGITAL ASSIGNMENT 1

a. Write a program to display "VIT University"

```
> display = function(){
+     print("VIT University")
+ }
> display()
[1] "VIT University"
```

b. Write a program to read two integers and display the result of addition.

```
> add = function(x,y){
+     return(x+y)
+ }
> x <- as.integer(readline(prompt="Enter number 1: "))
Enter number 1: 54
> y <- as.integer(readline(prompt="Enter number 2: "))
Enter number 2: 34
> add(x,y)
[1] 88
```

c. Write a program to perform arithmetic operations on a given set of numbers

```
> arithmetic.operations = function(x,y){
+     return(data.frame(X=x,Y=y,Sum=(x+y),Difference=(x-y),Product=(x*y),D
ivision=(x/y)))
+ }
> x=c(1,3,5,7,2)
> y=c(4,5,1,6,9)
> arithmetic.operations(x,y)
  X Y Sum Difference Product  Division
1 1 4   5         -3       4 0.2500000
2 3 5   8         -2      15 0.6000000
3 5 1   6          4       5 5.0000000
4 7 6  13          1      42 1.1666667
5 2 9  11         -7      18 0.2222222
```

d. Write a program to find the sum and average of marks of five subjects.

```
> marks = function(x){
+     sum_ = 0
+     for(i in x){
+         sum_ = sum_ + i
+     }
+     average_ = sum_/length(x)
+     return(c(Sum = sum_, Average = average_))
+ }
> student = c(44,56,78,36,87)
> marks(student)
    Sum Average
  301.0    60.2
```

e. Write a program to swap the values to two variables.

```
> swap = function(a,b){
+     temp = a
+     a = b
+     b = temp
+     return(c(A=a,B=b))
+ }
> swap(10,20)
 A  B
20 10
> x <- as.integer(readline(prompt="Enter number 1: "))
Enter number 1: 43
> y <- as.integer(readline(prompt="Enter number 2: "))
Enter number 2: 12
> swap(x,y)
 A  B
12 43
```

f. Write a program to display multiple lines

```
> display = function(){
+     cat("Hello \nVIT University\nVellore")
+ }
> display()
Hello
VIT University
Vellore
```

2. Simple arithmetic operations

a. Write a program to compute the area of a circle.

```
> circle = function(r){
+     return(pi*r*r)
+ }
> circle(3)
[1] 28.27433
>
> r=as.integer(readline(prompt="Enter the radius of a circle"))
Enter the radius of a circle6
> circle(r)
[1] 113.0973
```

b. Write a program to convert data from one form to another

i. Kilometer to Meter

```
> kmTom = function(x){
+     return(c(Kilometers=x,Meters=x*1000))
+ }
> kmTom(3)
Kilometers     Meters
         3       3000
>
>
> km <- as.integer(readline(prompt="Enter the kms: "))
Enter the kms: 6
> kmTom(km)
Kilometers     Meters
         6       6000
```

ii. Fahrenheit to Celsius

```
> FToC = function(f){
+     c = ((f-32)*5)/9
+     return(c(Fahrenheit=f,Celsius=c))
+ }
> FToC(50)
Fahrenheit    Celsius
        50         10
```

```
> f <- as.integer(readline(prompt="Enter the temperature in Fahrenheit: ")
)
Enter the temperature in Fahrenheit: 98
> FToC(f)
Fahrenheit    Celsius
  98.00000   36.66667
```

c. Write a program to swap the values of two integers without using a third variable.

```
> swap_2no = function(a,b){
+     a = a + b;
+     b = a - b;
+     a = a - b;
+     return(c(A=a,B=b))
+ }
> swap_2no(11,20)
 A  B
20 11
> x <- as.integer(readline(prompt="Enter number 1: "))
Enter number 1: 43
> y <- as.integer(readline(prompt="Enter number 2: "))
Enter number 2: 54
> swap_2no(x,y)
 A  B
54 43
```

d. Write a program to check whether the given number is even or odd.

```
> evenodd = function(x){
+     if(x%%2==0){
+         print("Number is Even")
+     }else{
+         print("Number is Odd")
+     }
+ }
> evenodd(2)
[1] "Number is Even"
> y <- as.integer(readline(prompt="Enter the number: "))
Enter the number: 13
> evenodd(y)
[1] "Number is Odd"
```

e. Write a program to check whether the input year is a leap year or not.

```
> year = function(x){
+     if(x%%4 == 0){
+         if( x%%100 == 0){
+             if(x%%400 == 0){
```

```
+                 print(paste(x," is a leap year."))
+             }else{
+                 print(paste(x," is not a leap year."))
+             }
+         }else{
+             print(paste(x," is a leap year."))
+         }
+     }else{
+         print(paste(x," is not a leap year."))
+     }
+ }
> year(2012)
[1] "2012  is a leap year."
> y <- as.integer(readline(prompt="Enter the year: "))
Enter the year: 2016
> year(y)
[1] "2016  is a leap year."
```

3. Usage of Control Structures

a. Write a program to find the greatest among the three given numbers

```
> greatest = function(x){
+     max_ = 0
+     for(i in 1:length(x)){
+         if(x[i]>max_){
+             max_ = x[i]
+         }
+     }
+     return(c(X=x,Greatest = max_))
+ }
> x = c(1,-3,2)
> greatest(x)
     X1       X2       X3 Greatest
      1       -3        2        2
```

b. Write a program to calculate energy bill. Read the starting and ending meter charges as given below.

```
> calculate = function(x,y){
+     reading = y-x
+     if(reading<100){
+         rate = reading*1.50
+     }else if(reading>=100 && reading<200){
+         rate = reading*2.50
+     }else if(reading>=200 && reading<=500){
+         rate = reading*3.50
+     }else if(reading>500){
+         rate = reading*5.00
```

```
+     }
+     return(c(Starting=x,Ending=y,Total_consumed=reading,Bill=rate))
+ }
> calculate(200,250)
      Starting         Ending Total_consumed           Bill
           200            250             50             75
>
> x <- as.integer(readline(prompt="Enter the starting reading: "))
Enter the starting reading: 102
> y <- as.integer(readline(prompt="Enter the ending reading: "))
Enter the ending reading: 306
> calculate(x,y)
      Starting         Ending Total_consumed           Bill
           102            306            204            714
```

2. MATRIX EXERCISES

Matrices Exercises 1

a) Suppose you want to create a vector x whose length is zero

```
> x <- vector(mode="numeric", length=0)
> x
numeric(0)
> length(x)
[1] 0
```

b) Create two 2x3 matrices.

c) Print them using print(matrix)

```
> x=matrix(c(2,3,1,7,5,3), nrow = 2, ncol = 3)
> print(x)
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    3    7    3
> y=matrix(c(8,7,3,5,2,4), nrow = 2, ncol = 3)
> print(y)
     [,1] [,2] [,3]
[1,]    8    3    2
[2,]    7    5    4
```

d) Multiply the matrices

```
> x*y
     [,1] [,2] [,3]
[1,]   16    3   10
[2,]   21   35   12
```

e) Divide the matrices

```
> x/y
          [,1]      [,2] [,3]
[1,] 0.2500000 0.3333333 2.50
[2,] 0.4285714 1.4000000 0.75
```

Exercise 2 Create three vectors x,y,z with integers and each vector has 3 elements. Combine the three vectors to become a 3×3 matrix A where each column represents a vector. Change the row names to a,b,c.

```
> x=c(1,3,6)
> y=c(3,2,7)
> z=c(6,4,2)
> A=cbind(x,y,z)
> A
     x y z
[1,] 1 3 6
[2,] 3 2 4
[3,] 6 7 2
> rownames(A)=c("a","b","c")
> A
  x y z
a 1 3 6
b 3 2 4
c 6 7 2
```

Exercise 3 Please check your result from Exercise 1, using is.matrix(A). It should return TRUE, if your answer is correct. Otherwise, please correct your answer. Hint: Note that is.matrix() will return FALSE on a non-matrix type of input. Eg: a vector and so on.

```
> is.matrix(A)
[1] TRUE
```

Exercise 4 Create a vector with 12 integers. Convert the vector to a 4*3 matrix B using matrix(). Please change the column names to x, y, z and row names to a, b, c, d. The argument byrow in matrix() is set to be FALSE by default. Please change it to TRUE and print B to see the differences.

```
> b=sample(1:50,12,replace = FALSE)
> m=matrix(b,nrow=4,ncol=3)
> rownames(m) =c("a","b","c","d")
> colnames(m)= c("x","y","z")
> m
   x  y  z
a 22 21 28
b 39 11  5
c 34 33  1
```

```
d 30 50 10
> m= matrix(b, nrow = 4, ncol = 3, byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]   22   39   34
[2,]   30   21   11
[3,]   33   50   28
[4,]    5    1   10
```

Exercise 5 Please obtain the transpose matrix of B named tB .

```
> tm=t(m)
> tm
     [,1] [,2] [,3] [,4]
[1,]   22   30   33    5
[2,]   39   21   50    1
[3,]   34   11   28   10
```

Exercise 6 Now tB is a 3×4 matrix. By the rule of matrix multiplication in algebra, can we perform tB*tB in R language? (Is a 3×4 matrix multiplied by a 3×4 allowed?) What result would we get?

```
> tm*tm
     [,1] [,2] [,3] [,4]
[1,]  484  900 1089   25
[2,] 1521  441 2500    1
[3,] 1156  121  784  100
```

This gives us the element wise multiplication of the matrices

```
> tm%*%tm
Error in tm %*% tm : non-conformable arguments
```

tB x tB would not be allowed because it disobeys the algebra rules for conventional matrix multiplication

Exercise 7 As we can see from Exercise 5, we were expecting that tB*tB would not be allowed because it disobeys the algebra rules. But it actually went through the computation in R. However, as we check the output result , we notice the multiplication with a single * operator is performing the component wise multiplication. It is not the conventional matrix multiplication. How to perform the conventional matrix multiplication in R? Can you compute matrix A multiplies tB ?

```
> A%*%tm
  [,1] [,2] [,3] [,4]
a  343  159  351   68
b  280  176  311   57
c  473  349  604   57
```

Exercise 8 If we convert A to a data.frame type instead of a matrix , can we still compute a conventional matrix multiplication for matrix A multiplies matrix A ? Is there any way we could still perform the matrix multiplication for two data.frame type variables? (Assuming proper dimension)

```
> A = data.frame(A)
> as.matrix(A)%*%as.matrix(A)
   x  y  z
a 46 51 30
b 33 41 34
c 39 46 68
```

Exercise 9 Extract a sub-matrix from B named subB . It should be a 3×3 matrix which includes the last three rows of matrix B and their corresponding columns.

```
> b = sample(1:50,12,replace = FALSE)
> B = matrix(b, nrow = 4, ncol = 3, byrow = TRUE)
>
> subB=B[2:dim(B)[1],1:3]
> subB
     [,1] [,2] [,3]
[1,]   29   22   47
[2,]    8   43   50
[3,]   41    2   44
```

Exercise 10 Compute 3*A , A+subB , A-subB . Can we compute A+B? Why?

```
> 3*A
   x  y  z
a  3  9 18
b  9  6 12
c 18 21  6
> A+subB
   x  y  z
a 30 25 53
b 11 45 54
c 47  9 46
> A-subB
    x   y   z
a -28 -19 -41
b  -5 -41 -46
c -35   5 -42
```

Exercise 11 Generate a n * n matrix (square matrix) A1 with proper number of random numbers, then generate another n * m matrix A2. If we have A1*M=A2 (Here * represents the conventional multiplication), please solve for M. Hint: use the runif() and solve() functions. E.g., runif(9) should give you 9 random numbers.

```
> A1 = matrix(runif(9),3,3)
> A2 = matrix(runif(12),3,4)
> M=solve(A1,A2)
> M
           [,1]      [,2]       [,3]        [,4]
[1,]  1.635468 0.2387663  0.9735393  0.07292297
[2,]  1.049999 0.5407275  0.3752523  1.09538621
[3,] -2.627461 0.1970246 -0.4410717 -1.07313373

>
```

1) Suppose X is a matrix in R. Which of the following is not equivalent to X?

A) t(t(X))

B) X %*% matrix(1,ncol(X))

C) X*1

D) X%*%diag(ncol(X))

```
> X
     [,1] [,2] [,3]
[1,]   15   40    5
[2,]   29   22   47
[3,]    8   43   50
[4,]   41    2   44
> t(t(X))
     [,1] [,2] [,3]
[1,]   15   40    5
[2,]   29   22   47
[3,]    8   43   50
[4,]   41    2   44
> X %*% matrix(1,ncol(X) )
     [,1]
[1,]   60
[2,]   98
[3,]  101
[4,]   87
> X*1
     [,1] [,2] [,3]
[1,]   15   40    5
[2,]   29   22   47
[3,]    8   43   50
[4,]   41    2   44
> X%*%diag(ncol(X))
     [,1] [,2] [,3]
```

```
[1,]   15   40    5
[2,]   29   22   47
[3,]    8   43   50
[4,]   41    2   44
```

As we can see option B is not equivalent to X

2) Load the following two matrices into R: a <- matrix(1:12, nrow=4) b <- matrix(1:15, nrow=3) In the question below, we will use the matrix multiplication operator in R, %*%, to multiply these two matrices. What is the value in the 3rd row and the 2nd column of the matrix product of a and b?

```
> a <- matrix(1:12, nrow=4)
> b <- matrix(1:15, nrow=3)
> c=a%*%b
> c[3,2]
[1] 113
```

3) Using the the diag function build a diagonal matrix of size 4 with the following values in the diagonal 4,1,2,3.

```
> diag(x = c(4,1,2,3), nrow = 4, ncol = 4)
     [,1] [,2] [,3] [,4]
[1,]    4    0    0    0
[2,]    0    1    0    0
[3,]    0    0    2    0
[4,]    0    0    0    3
```

4)Write code for displaying Identity Matrix (Note: a square matrix in which all the elements of the principal diagonal are ones and all other elements are zeros)

```
> diag(3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> diag(4)
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

5)Write a code to display 3 x 1 zero matrix (Note: A zero matrix is an matrix consisting of all 0s)

```
> z=matrix(0,3,1)
> z
     [,1]
[1,]    0
[2,]    0
[3,]    0
```

7) Write a code to display Inverse of a Matrix

```
> A
  x y z
a 1 3 6
b 3 2 4
c 6 7 2
> solve(A)
           a          b             c
x -0.2857143  0.4285714 -7.401487e-17
y  0.2142857 -0.4047619  1.666667e-01
z  0.1071429  0.1309524 -8.333333e-02
```

8) Write a code to display Inverse & Determinant of a Matrix

```
> A
     x y z
[1,] 1 3 6
[2,] 3 2 4
[3,] 6 7 2
> solve(A)
        [,1]       [,2]          [,3]
x -0.2857143  0.4285714 -7.401487e-17
y  0.2142857 -0.4047619  1.666667e-01
z  0.1071429  0.1309524 -8.333333e-02
> det(A)
[1] 84
```

9) Write a code to find dimensions of a Matrix.

```
> dim(A)
[1] 3 3
```

10) Consider the following matrix A <- matrix(c(2,3,-2,1,2,2),3,2) . Find the sum of columns and rows

```
> A <- matrix(c(2,3,-2,1,2,2),3,2)
> A
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
> colSums(A)
[1] 3 5
> rowSums(A)
[1] 3 5 0
```

3. FACTOR EXERCISES

1) If x = c(1, 2, 3, 3, 5, 3, 2, 4, NA), what are the levels of factor(x)?

```
> x = c(1, 2, 3, 3, 5, 3, 2, 4, NA)
> factor(x)
[1] 1    2    3    3    5    3    2    4    <NA>
Levels: 1 2 3 4 5
```

2) Let x <- c(11, 22, 47, 47, 11, 47, 11). If an R expression factor(x, levels=c(11, 22, 47), ordered=TRUE) is executed, what will be the 4th element in the output?

```
> x <- c(11, 22, 47, 47, 11, 47, 11)
> c=factor(x, levels=c(11, 22, 47), ordered=TRUE)
> c[4]
[1] 47
Levels: 11 < 22 < 47
```

3) If z <- c("p", "a" , "g", "t", "b"), then which of the following R expressions will replace the third element in z with "b".

a. factor(z[3]) <- "b"

b. levels(z[3]) <- "b"

c. z[3] <- "b"

```
> factor(z[3]) = "b"
Error in factor(z[3]) = "b" : could not find function "factor<-"
> levels(z[3]) <- "b"
> z
[1] "p" "a" "g" "t" "b"
> z[3] <- "b"
> z
[1] "p" "a" "b" "t" "b"
```

The third option changes the value.

4) Create a factor x <- factor(c("single", "married", "married", "single")) find out the 3rd element.

```
> x <- factor(c("single", "married", "married", "single"))
> x[3]
[1] married
Levels: married single
```

5) Find out the 2nd and 4th element.

```
> x[2]
[1] married
Levels: married single
> x[4]
[1] single
Levels: married single
```

6) Find out all the element except the 1st. Consider the factor responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree")), with the following output: [1] Agree Agree Strongly Agree Disagree Agree Levels: Agree Disagree Strongly Agree Later it was found that new a level "Strongly Disagree" exists. Write an R expression that will Include "strongly disagree" as new level attribute of the factor and returns the following output: [1] Agree Agree Strongly Agree Disagree Agree Levels: Strongly Agree Agree Disagree Strongly Disagree

```
> responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "A
gree"))
> responses
[1] Agree          Agree          Strongly Agree
[4] Disagree       Agree
Levels: Agree Disagree Strongly Agree
> factor(responses, levels=c("Strongly Agree", "Agree", "Disagree", "Stron
gly Disagree"))
[1] Agree          Agree          Strongly Agree
[4] Disagree       Agree
4 Levels: Strongly Agree Agree ... Strongly Disagree
```

7) Let x <- data.frame(q=c(2, 4, 6), p=c("a", "b", "c")). Write an R statement that will replace levels a,b, c with labels "fertiliser1", "fertliser2", "fertiliser3".

```
> x <- data.frame(q=c(2, 4, 6), p=c("a", "b", "c"))
```

```
> x$p <- factor(x$p, levels=c("a", "b", "c"), labels=c("fertiliser1", "fer
tiliser2", "fertiliser3"))
> x
  q          p
1 2 fertiliser1
2 4 fertiliser2
3 6 fertiliser3
```

8) Consider 10 matrix exercises of your choice and solve with R Programming(Write problem statement correctly)

a) Consider A=matrix(c(2,0,1,3), ncol=2) and B=matrix(c(5,2,4,-1), ncol=2).
a) Find A + B
b) Find A – B

```
> A <- matrix(c(2,0,1,3), ncol=2)
> B <- matrix(c(5,2,4,-1),ncol=2)
>
> A+B
     [,1] [,2]
[1,]    7    5
[2,]    2    2
> A-B
     [,1] [,2]
[1,]   -3   -3
[2,]   -2    4
```

b) Scalar multiplication. Find the solution for aA where a=3 and A is the same as in the previous question.

```
> a <- 3
> a*A
     [,1] [,2]
[1,]    6    3
[2,]    0    9
```

c) Using the the diag function build a diagonal matrix of size 4 with the following values in the diagonal 4,1,2,3.

```
> diag(4)*c(4,1,2,3)
     [,1] [,2] [,3] [,4]
[1,]    4    0    0    0
[2,]    0    1    0    0
[3,]    0    0    2    0
[4,]    0    0    0    3
```

d) Find the solution for Ab, where A is the same as in the previous question and b=c(7,4).

```
> b <- c(7,4)
> b%*%A
     [,1] [,2]
[1,]   14   19
```

e) Find the solution for AB, where B is the same as in question 1.

```
> A%*%B
     [,1] [,2]
[1,]   12    7
[2,]    6   -3
```

f) Find the transpose matrix of A.

```
> t(A)
     [,1] [,2]
[1,]    2    0
[2,]    1    3
```

g) Find the inverse matrix of A.

```
> solve(A)
     [,1]       [,2]
[1,]  0.5 -0.1666667
[2,]  0.0  0.3333333
```

h) Find the value of x on Ax=b.

```
> solve(A,b)
[1] 2.833333 1.333333
```

i) Using the function eigen find the eigenvalue for A.

```
> eigen(A)$values
[1] 3 2
```

j) Find the eigenvalues and eigenvectors of A'A . Hint: Use crossprod to compute A'A .

```
> eigen(crossprod(A))
eigen() decomposition
$`values`
[1] 10.605551  3.394449

$vectors
          [,1]       [,2]
[1,] 0.2897841 -0.9570920
[2,] 0.9570920  0.2897841
```

9) Consider 10 factor exercises of your choice and solve with R(Write problem statement correctly).

a) If z <- factor(c("p", "q", "p", "r", "q")) and levels of z are "p", "q" ,"r", write an R expression that will change the level "p" to "w" so that z is equal to: "w", "q" , "w", "r" , "q".

```
> z <- factor(c("p", "q", "p", "r", "q"))
> levels(z)[1] <- "w"
> z
[1] w q w r q
Levels: w q r
```

b) If:
s1 <- factor(sample(letters, size=5, replace=TRUE)) and
s2 <- factor(sample(letters, size=5, replace=TRUE)),
write an R expression that will concatenate s1 and s2 in a single factor with 10 elements.

```
> s1 <- factor(sample(letters, size=5, replace=TRUE))
> s2 <- factor(sample(letters, size=5, replace=TRUE))
> factor(c(levels(s1)[s1], levels(s2)[s2]))
 [1] l e k r i f w a p y
Levels: a e f i k l p r w y
```

c) Consider the iris data set in R. Write an R expression that will 'cut' the Sepal.Length variable and create the following factor with five levels.

(4.3, 5.02] (5.02, 5.74] (5.74, 6.46] (6.46, 7.18] (7.18, 7.9]
32 41 42 24 11

```
> table(cut(iris$Sepal.Length, 5))

 (4.3,5.02] (5.02,5.74] (5.74,6.46] (6.46,7.18]  (7.18,7.9]
         32          41          42          24          11
```

d) If x <- factor(c("high", "low", "medium", "high", "high", "low", "medium")), write an R expression that will provide unique numeric values for various levels of x with the following output:

levels value
1 high 1
2 low 2
3 medium 3

```
> x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))
> data.frame(levels = unique(x), value = as.numeric(unique(x)))
  levels value
1   high     1
2    low     2
3 medium     3
```

e) Write a R program to create a factor corresponding to height of women data set, which contains height and weights for a sample of women.

```
> data = women
> print(data)
   height weight
1      58    115
2      59    117
3      60    120
4      61    123
5      62    126
6      63    129
7      64    132
8      65    135
9      66    139
10     67    142
11     68    146
12     69    150
13     70    154
14     71    159
15     72    164
> height_f = cut(women$height,3)
> print(table(height_f))
height_f
  (58,62.7] (62.7,67.3]   (67.3,72]
          5           5           5
> weight_f = cut(women$weight,3)
> print(table(weight_f))
weight_f
(115,131] (131,148] (148,164]
        6         5         4
```

f) Write a R program to convert a given pH levels of soil to an ordered factor.

```
> ph = c(1,3,10,7,5,4,3,7,8,7,5,3,10,10,7)
> ph
 [1]  1  3 10  7  5  4  3  7  8  7  5  3 10 10  7
> ph_f = factor(ph,levels=c(3,7,10),ordered=TRUE)
> ph_f
 [1] <NA> 3    10   7    <NA> <NA> 3    7    <NA> 7    <NA>
[12] 3    10   10   7
Levels: 3 < 7 < 10
```

g) Write a R program to concatenate two given factor in a single factor.

```
> f1 <- factor(sample(LETTERS, size=6, replace=TRUE))
> f2 <- factor(sample(LETTERS, size=6, replace=TRUE))
> f1
[1] Z E I N P D
Levels: D E I N P Z
> f2
[1] H E V K T W
Levels: E H K T V W
> f = factor(c(levels(f1)[f1], levels(f2)[f2]))
> f
 [1] Z E I N P D H E V K T W
Levels: D E H I K N P T V W Z
```

h) Write a R program to create an ordered factor from data consisting of the names of months.

```
> mons = c("March","April","January","November","January",
+            "September","October","September","November","Augus
t","February",
+            "January","November","November","February","May","A
ugust","February",
+            "July","December","August","August","September","No
vember","September",
+            "February","April")
> mons
 [1] "March"     "April"     "January"   "November"
 [5] "January"   "September" "October"   "September"
 [9] "November"  "August"    "February"  "January"
[13] "November"  "November"  "February"  "May"
[17] "August"    "February"  "July"      "December"
[21] "August"    "August"    "September" "November"
[25] "September" "February"  "April"
> f=factor(mons)
> f
 [1] March     April     January   November  January
 [6] September October   September November  August
[11] February  January   November  November  February
[16] May       August    February  July      December
[21] August    August    September November  September
```

```
[26] February  April
11 Levels: April August December February January ... September
> table(f)
f
    April    August  December  February   January      July
        2         4         1         4         3         1
    March       May  November   October September
        1         1         5         1         4

>
```

i) Write a R program to find the levels of factor of a given vector.

```
> v = c(1, 2, 3, 3, 4, NA, 3, 2, 4, 5, NA, 5)
> v
 [1]  1  2  3  3  4 NA  3  2  4  5 NA  5
> levels(factor(v))
[1] "1" "2" "3" "4" "5"
```

j) Consider again the iris data set. Write an R expression that will generate a two-way frequency table with two rows and three colums. The rows should relate to Sepal.length (less than 5: TRUE or FALSE) and columns to Species, with the following output:

setosa versicolor virginica
FALSE 30 49 49
TRUE 20 1 1

```
> table(iris$Sepal.Length < 5, factor(iris$Species))

       setosa versicolor virginica
  FALSE     30         49        49
  TRUE      20          1         1

>
```

4. Control Statement (looping, conditional statements)

1. Houseflies have approximately age of four weeks. Give the number of days a housefly lived, the R code to determine its approximate age in seconds. Check for boundary conditions and print 'Invalid input' if condition is not satisfied. For example, if a housefly lived for 21 days then its approximate age in seconds is 24*21*60*60 is 1814400.

```
> calculate=function(x){
+ if(x>28 || x<=0){
+ print('Invalid input')}
+ else{
+ s=x*24*60*60}
+ return(s)}
> n=as.integer(readline(prompt=("Enter the number of days housefly has liv
ed")))
Enter the number of days housefly has lived9
> calculate(n)
[1] 777600
```

2. Given two numbers ‘m’ and ‘n’, write the R code to check if they are relatively prime. Two integers ‘a’ and ‘b’ are said to be relatively prime, mutually prime, or coprime, if the only positive integer that divides both of them is 1. For example, 14 and 15 are coprime since the factors of 14 are 1, 2, 7, and 14 and factors of 15 are 1, 3, 5, and 15 and there is no common factor other than 1. (Use only conditional and iterational statements for designing the algorithm and implementation).

```
> calculate=function(x,y){
+   t=min(x,y)
+   flag=0
+   i=0
+   for(i in 2:t){
+     if(x%%i==0 && y%%i==0){
+       flag=1}
+
+     }
+   if(flag==1){
+    print("not coprime")}
+    else{
+    print("coprime")
+     }
+    }
> calculate(14,7)
[1] "not coprime"
> calculate(14,15)
[1] "coprime"
```

3. A city has a dam of capacity ‘x’ litres, water comes to the dam from ‘n’ places. Given the value of ‘n’ and the quantity of water (In litres and millilitres) that comes from ‘n’ places, write the R code to determine the total amount of water in the dam. Assume that the total quantity of water will be always less than the capacity of the dam. For example, if there are three places from which water comes to the dam and the water from place 1 is 2 litres 500 ml, water from second place is 3 litres 400 ml and water from third place is 1 litre 700 ml then the total quantity of water in dam will be 7 litres 600 ml.

```
> n=as.integer(readline())
```

```
3
> lit=0
> ml=0
> for(i in 1:n){
+     lit=lit+as.integer(readline())
+     ml=ml+as.integer(readline())
+ }
4
500
3
600
3
700
> print(lit+(ml%/%1000))
[1] 11
> print(ml%%1000)
[1] 800
```

4. Given ‘n’, the number of rows, design an algorithm and write an R code to draw a pattern. If n is ‘5’, the pattern looks as shown below:

```
*
**
****
******
********
```

```
> n=as.integer(readline())
7
> for(i in 1:n){
+     for(j in 1:i){
+
+         cat(paste("*"))
+     }
+
+     cat(paste("\n"))}
*
**
***
****
*****
******
*******
```

5. A. Consider 10 problems of your choice (mention problem statements correctly) related to loop using R

a) Write a for loop that iterates over the numbers 1 to 7 and prints the cube of each number using print().

```
> for (i in 1:7) {
+     print(i^3)
+ }
[1] 1
[1] 8
[1] 27
[1] 64
[1] 125
[1] 216
[1] 343
```

b) Write a for loop that iterates over the column names of the inbuilt iris dataset and print each together with the number of characters in the column name in parenthesis. Example output: Sepal.Length (12). Use the following functions print(), paste0() and nchar().

```
> for (n in names(iris)) {
+     print(paste0(n, " (", nchar(n), ")"))
+ }
[1] "Sepal.Length (12)"
[1] "Sepal.Width (11)"
[1] "Petal.Length (12)"
[1] "Petal.Width (11)"
[1] "Species (7)"
```

c) Write a while loop that prints out standard random normal numbers (use rnorm()) but stops (breaks) if you get a number bigger than 1.

```
> set.seed(3)
>
> while (TRUE) {
+     x <- rnorm(1)
+     print(x)
+     if (x > 1) {
+         break
+     }
+ }
[1] -0.9619334
[1] -0.2925257
[1] 0.2587882
[1] -1.152132
[1] 0.1957828
[1] 0.03012394
[1] 0.08541773
[1] 1.11661

>
```

d) Using next adapt the loop from last exercise so that doesn't print negative numbers.

```
while (TRUE) {
  x <- rnorm(1)
  if (x < 0) {
    next
  }
  print(x)
  if (x > 1) {
    break
  }
}
```

e) Using a for loop simulate the flip a coin twenty times, keeping track of the individual outcomes (1 = heads, 0 = tails) in a vector that you preallocte.

```
> n <- 20
> coin_outc <- vector(length = n, mode = "integer")
> for (i in 1:20) {
+     coin_outc[i] <- sample(c(0L, 1L), 1)
+ }
> coin_outc
 [1] 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
```

f) Use a nested for loop (a for loop inside a for loop) that produces the following matrix, preallocate the matrix with NA values.

```
> mat <- matrix(NA_integer_, nrow = 5, ncol = 5)
> for (i in 1:5) {
+     for (j in 1:5) {
+         mat[i, j] <- abs(i - j)
+     }
+ }
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    2    3    4
[2,]    1    0    1    2    3
[3,]    2    1    0    1    2
[4,]    3    2    1    0    1
[5,]    4    3    2    1    0
```

g) Use a while loop to investigate the number of terms required before the product

1·2·3·4·

reaches above 10 million.

```
> target <- 1e8
```

```
> product <- 1
> n <- 0
> while (product <= target) {
+     n <- n + 1
+     product <- product*n
+ }
> n
[1] 12
```

h) Use a while loop to simulate one stock price path starting at 100 and random normally distributed percentage jumps with mean 0 and standard deviation of 0.01 each period. How long does it take to reach above 150 or below 50?

```
> price_start <- 100
> price <- price_start
> target_diff <- 50
> n <- 0
> while (abs(price_start - price) < target_diff) {
+     n <- n + 1
+     price <- price*(1 + rnorm(mean = 0, sd = 0.01, n = 1))
+ }
> n
[1] 4292
```

i) Implement a simple version of Guess the number game using a while loop. The user should guess a number between 1 and 10, you can use scan() to get user input. The loop should break if the user guesses 5.

```
> x <- 0L
> while (TRUE) {
+     cat(
+         "I am thinking of a number between 1 and 10.",
+         "\nTake a guess and press enter twice!"
+     )
+     x <- scan()[1]
+     if (x == 5) {
+         cat("Right!")
+         break
+     }
+     cat("Wrong\n")
+ }
I am thinking of a number between 1 and 10.
Take a guess and press enter twice!
1: 3
2:
Read 1 item
Wrong
I am thinking of a number between 1 and 10.
Take a guess and press enter twice!
1: 7
2:
Read 1 item
```

```
Wrong
I am thinking of a number between 1 and 10.
Take a guess and press enter twice!
1: 3
2:
Read 1 item
Wrong
I am thinking of a number between 1 and 10.
Take a guess and press enter twice!
1: 8
2:
Read 1 item
Wrong
I am thinking of a number between 1 and 10.
Take a guess and press enter twice!
1: 5
2:
Read 1 item
Right!
```

j) Implement a multiplication game. A while loop that gives the user two random numbers from 2 to 12 and asks the user to multiply them. Only exit the loop after five correct answers. Try using as.integer(readline()) instead of scan() this time.

```
> total <- 0
> while (TRUE) {
+     if (!total %in% 0:4) {
+         total <- 0L
+     }
+     n <- sample(c(2:12), 2)
+     cat("What is the product of ", n[1], " and ", n[2], "?\n",
sep = "")
+     x <- as.integer(readline())
+     if (x == prod(n)) {
+         total <- total + 1L
+         cat("Right!")
+         if (total == 5) break
+         cat(" You just need", 5 - total, "more answers.\n\n")
+     } else {
+         cat("Wrong\n")
+     }
+ }
What is the product of 4 and 8?
32
Right! You just need 4 more answers.

What is the product of 6 and 10?
60
Right! You just need 3 more answers.

What is the product of 2 and 9?
18
Right! You just need 2 more answers.

What is the product of 5 and 7?
35
Right! You just need 1 more answers.

What is the product of 5 and 12?
60
```

```
Right!

>
```

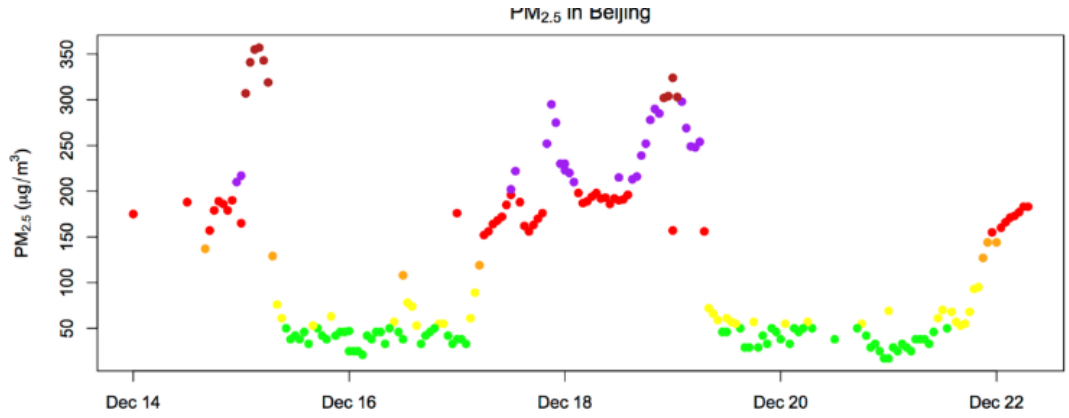
B. 10 problems of your choice (mention problem statements correctly) related to (if, if-else) concept and solve them using R

a) Based on the below generate a grade

"F" if final_score <60
"D" if 60≤final_score<70
"C" if 70≤final_score<80
"B" if 80≤final_score<90
"A" if 90≤final_score

```
> final_score=as.integer(readline())
78
> grade<-ifelse(final_score<60,"F", ifelse (final_score<70,"D", ifelse(fin
al_score<80,"C", ifelse (final_score<90,"B", "A"))))
> grade
[1] "C"
```

b) This type of scheme is really useful for putting different colors in graphs for different conditions. Consider the following graph of air quality measurements taken hourly in Beijing.



Give the color scheme based on the pm25 entered

```
> pm25=as.integer(readline())
170
> col = ifelse(pm25<=50,"green",
+
+              ifelse(pm25<101,"yellow",
```

```
+
+                     ifelse(pm25<150,"orange",
+
+                            ifelse(pm25<201,"red",
+
+                                   ifelse(pm25<301,"purple",
+
+                                          "firebrick")
+
+                            )
+
+                     )
+
+              )
+
+ )
>
>
> col
[1] "red"
```

c) Create an R script that calculates the square root of a given integer vector x of length one, if the value contained in x is negative it should return NA.

```
> x <- 16
> y <- ifelse(x >= 0, x, NA)
> cat("The square root of",  x, "is", sqrt(y))
The square root of 16 is 4
```

d) Create an R script that returns the maximum value out of the elements of a numeric vector x of length 2.

```
> x <- c(10, 1)
> if(x[1] > x[2]) {
+     cat("Max value is", x[1], "\n")
+ } else cat("Max value is", x[2], "\n")
Max value is 10
```

e) Create an R script that returns TRUE if the elements of a vector x, with length 3, are strictly increasing

```
> x <- c(10, 11, 12)
> grow <- FALSE
>
> ifelse ( ( (x[1] < x[3] & x[1] < x[2]) & x[2] < x[3]), grow <- TRUE, gro
w)
[1] TRUE
> if (grow){
+     cat ("Increasing strictly \n")
```

```
+ } else cat ("Not increasing strictly \n")
Increasing strictly
```

f) Create an R script that returns the max value of a vector x with length 3. Don't use the aid of an auxiliary variable.

```
> x <- c(20, 10, 1)
>
> if (x[1] > x[2] & x[1] > x[3] ) {
+     cat (x[1], "\n" )
+ } else if (x[2] > x[3] ) {
+     cat (x[2] , "\n" )
+ } else {
+     cat (x[3] , "\n" )
+ }
20
```

g) Create an R script that returns the amount of values that are larger than the mean of a vector. You are allowed to use mean().

```
> x <- c(-100, 10, 20, 30, 50, 51, 52, 53, 54, 55)
> counter <- 0
> mean <- mean(x)
>
> for (i in 1:length(x)){
+
+     if(x[i] > mean){
+         counter <- counter +1
+     }
+ }
>
> cat("The number of values that are bigger than the mean is", counter, "\
n")
The number of values that are bigger than the mean is 7
```

h) Create an R script that, given a numeric vector x with length 3, will print the elements by order from high to low.

```
> x <- c(30, 120, 100)
>
> if (x[1] > x[2]){
+     fir <- x[1]
+     sec <- x[2]
+ } else {
+     fir <- x[2]
```

```
+     sec <- x[1]
+ }
> if ( x[3] > fir & x[3] > sec ) {
+     thi <- sec
+     sec <- fir
+     fir <- x[3]
+ } else if ( x[3] < fir & x[3] < sec ) {
+     thi <- x[3]
+ } else {
+     thi <- sec
+     sec <- x[3]
+ }
> cat (fir, sec, thi, "\n")
120 100 30
```

i) Count the number of integers which are even in a vector

```
> x <- c(2,5,3,9,8,11,6)
>
> count <- 0
>
> for (val in x)
+
+ {
+
+     if(val %% 2 == 0)  count = count+1
+
+ }
>
> print(count)
[1] 3
```

j) The numeric system represented by Roman numerals originated in ancient Rome and remained the usual way of writing numbers throughout Europe well into the late Middle Ages. Numbers in this system are represented by combinations of letters as shown below:

Roman Numeral	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Given a letter in Roman numeral, develop an algorithm and write the Python code to print the value of it. If some other letter other than Roman numeral is given as input then print 'Enter a roman numeral' and terminate.

```
> r=readline()
V
```

```
> if(r=="l"){
+   cat(paste(1))
```

```
+ }else if(r=='V'){
+   print(5)
+   }else if(r == 'X'){
+   print(10)
+   }else if(r == 'L'){
+   print(50)
+   }else if(r == 'C'){
+   print(100)
+   }else if(r == 'D'){
+   print(500)
+   }else if(r == 'M'){
+   print(1000)
+   }else{
+   print("Enter a roman numeral")}
[1] 5
```