R Programming Language

Module 1& 2

Program 1: Vector Exercises

1. **Create empty vector and append values.**

```
# create an empty vector a
a=c()
# display it
print(a)
# adding numbers from 1 to
# 20 to a vector
a=1:20
```

2. **Find Sum, Mean and Product of a Vector in R**

```
vec = c(1, 2, 3 , 4)
print("Sum of the vector:")
# inbuilt sum method
print(sum(vec))
# using inbuilt mean method
print("Mean of the vector:")
print(mean(vec))
# using inbuilt product method
print("Product of the vector:")
print(prod(vec))
```

3. **Find product of vector elements in R**

```
# declaring a floating point vector
vec <- c(1.1,2,3.2,4)
size = length(vec)
prod = 1
for(i in 1:size)
{
   prod = vec[i]*prod
}
print("Product of vector elements:")
print(prod)
```

4. **Count the specific value in a given vector in R**

```
x = c(10, 20, 30, 20, 20, 25, 9, 26)
print("Original Vectors:")
print(x)
key=as.integer(readline(“enter a number to be count”))
print(sum(x==key))
```

5. Remove Multiple Values from Vector in R

```
# create a vector
a=c(1,2,"Joe",4,5,"Bobby",4,5,6,"Joy","Rohith",56.0)
print(a)
# Remove multiple values
a <- a[! a % in% c("Joe",4,6, "Joy")]
# display a
print(a)
```

Note: The %in% operator **in R** is used to check if the values of the first argument are present in the second argument and returns a logical vector

Functions Exercises

1. Find Factorial of a number using recursion

```
fact<- function(n)
{
  if(n==0)
    return(TRUE)
  else
    return(fact(n-1)*n)
}
n=as.integer(readline(“enter the value:”))
result=fact(n)
print(result)
```

2. Find the Fibonacci Sequence Using Recursive Function

```
# take input from the user
n = as.integer(readline(prompt="How many terms? "))
# check if the number of terms is valid
if(n <= 0) {
  print("Please enter a positive integer")
} else {
  print("Fibonacci sequence:")
  for(i in 0:(n-1)) {
    print(fib(i))
  }
}
```

3. Sum of Series Using Recursion

```
sum <- function(vec){
  if(length(vec)<=1){
    return(vec^2)
  }else{
    return(vec[1]^2+sum(vec[-1]))
  }
}
n<- c(1:5)
result=sum(n)
print(result)
```

4.selection Sort

```
selection<-function(arr)
{
  n<-length(arr)
  for(i in 1:(n-1))
  {
    for(j in (1+i):(n))
    {
      if(arr[j]<arr[i])
      {
        temp=arr[i]
        arr[i]=arr[j]
        arr[j]=temp
      }
    }
  }
  arr
}
arr<- sample(1:100,10)
sort<- selection(arr)
print(sort)
```

5.Bubble sort:

```
bubblesort<-function(arr)
{
  n<-length(arr)

  for(i in 1:(n-1))
  {
    for(j in (1):(n-i))
    {

      if(arr[j]>arr[j+1])
      {
        temp=arr[j]
        arr[j]=arr[j+1]
        arr[j+1]=temp
      }
    }
  }
  arr
}
arr<- sample(1:100,10)
sort<- bubblesort(arr)
print(sort)
```

3. Matrix Exercises

1. Create the matrix

$$A = \left| \begin{matrix} 1 & 7 & 3 \\ 4 & 4 & 6 \\ 4 & 7 & 12 \end{matrix} \right.$$

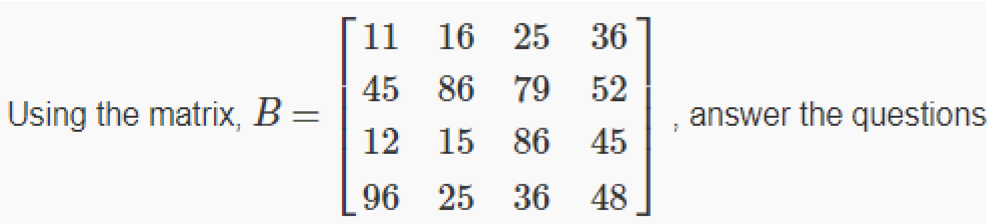
   a. Change the element 12 to 13.
   b. Access the second row and the third column.
   c. List all the elements in the second column and third row.
   d. How do you access the sub-matrix

$$\begin{matrix} 1 & 3 \\ 4 & 6 \end{matrix}$$

Program:

```
A<-matrix(c(1,7,3,4,4,6,4,7,12),nrow=3, ncol=3, byrow=TRUE,dimnames=list(c("p","q","r"),c("x","y","z")))
print(A)            # Display the matrix
A[3,3]<- 13       # Changing the element
print(A)
second_col<-A[c("p","q","r"),c("y")] # List all elements of second column
print(second_col)
third_row<-A[c("r"),c("x","y","z")]                          # List all elements of third row
print(third_row)
sub_matrix<-A[c("p","q"),c("x","z")]                         # Access the sub-matrix
print(sub_matrix)
```

2. .



Using the matrix, $B = \begin{bmatrix} 11 & 16 & 25 & 36 \\ 45 & 86 & 79 & 52 \\ 12 & 15 & 86 & 45 \\ 96 & 25 & 36 & 48 \end{bmatrix}$, answer the questions

a. Display the full matrix **B**?

b. What is the expected output when the command B[1,3]?

c. Add a fifth column:

10
11
12
13

d. What is the command to exclude the elements of 3rd row and select the rest of matrix?

a. Program with Output:

```
B <- matrix(c(11, 16, 25, 36, 45, 86, 79, 52, 12, 15, 86, 45, 96, 25, 36, 48), nrow = 4, ncol = 4, byrow = TRUE)
>
> # a. Display the full matrix B
> B
     [,1] [,2] [,3] [,4]
[1,]   11   16   25   36
[2,]   45   86   79   52
[3,]   12   15   86   45
[4,]   96   25   36   48
>
> # b. Output of B[1,3]
> B[1, 3]
[1] 25
>
> # c. Add a fifth column [10 11 12 13]
> B <- cbind(B, c(10, 11, 12, 13))
>
> # d. Exclude the elements of the 3rd row and select the rest of the matrix
> X<- B[-3, ]
> X
     [,1] [,2] [,3] [,4] [,5]
[1,]   11   16   25   36   10
[2,]   45   86   79   52   11
[3,]   96   25   36   48   13
```

3. Create two matrices A and B.

$$A = \begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{matrix} \qquad \text{and} \qquad B = \begin{matrix} 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 \end{matrix}$$

Determine the following:

a. A+B

b. A-B

c. A*B – regular matrix multiplication

d. A*B – element-wise matrix multiplication

e. A/B – element-wise matrix division

Program with Output:

```
# Create matrix A
> A <- matrix(c(1:16), nrow = 4, ncol = 4, byrow = TRUE)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16

> # Create matrix B
> B <- matrix(c(17:32), nrow = 4, ncol = 4, byrow = TRUE)
> B
     [,1] [,2] [,3] [,4]
[1,]   17   18   19   20
[2,]   21   22   23   24
[3,]   25   26   27   28
[4,]   29   30   31   32

> # a. A + B
> result_sum <- A + B
> result_sum
     [,1] [,2] [,3] [,4]
[1,]   18   20   22   24
[2,]   26   28   30   32
[3,]   34   36   38   40
[4,]   42   44   46   48
```

```
> # b. A - B
> result_diff <- A - B
> result_diff

      [,1]  [,2]  [,3]  [,4]
[1,]  -16  -16  -16  -16
[2,]  -16  -16  -16  -16
[3,]  -16  -16  -16  -16
[4,]  -16  -16  -16  -16
```

```
> # c. A * B - regular matrix multiplication
> result_mult_reg <- A %*% B
> result_mult_reg
       [,1]    [,2]    [,3]    [,4]
[1,]   250     260     270     280
[2,]   618     644     670     696
[3,]   986     1028  1070  1112
[4,]  1354    1412 1470  1528
```

```
> # d. A * B - element-wise matrix multiplication
> result_mult_elem <- A * B
> result_
    mult_e
    lem
    [,1]
          [
    ,2]  [,3]
    [,4]
[1,]   17   36    57    80
[2,]   105 132  161  192
[3,]   225 260  297  336
[4,]   377 420  465  512
```

```
> # e. A / B - element-wise matrix division
> result_div_elem <- A / B
> result_div_elem
         [,1]            [,2]          [,3]          [,4]
[1,]  0.05882353   0.1111111   0.1578947   0.2000000
[2,]  0.23809524   0.2727273   0.3043478   0.3333333
[3,]  0.36000000   0.3846154   0.4074074   0.4285714
[4,]  0.44827586   0.4666667   0.4838710   0.5000000
```