**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**SAVEETHA SCHOOL OF ENGINEERING**

**LAB RECORD**

**ITA0443 - Statistics with R Programming**

**For**

**Real – time Problems**

**Prakash.D**

**191911051**

**Github Link:** [**https://github.com/prakashd007/R-programming-lab-record**](https://github.com/prakashd007/R-programming-lab-record)

**LIST OF EXPERIMENTS**

1. BASIC OPERATIONS IN R
2. DATA STRUCTURES IN R
3. WORKING WITH LOOPING & FUNCTION IN R
4. IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY & RECURSION

**1.BASIC OPERATIONS IN R**

Exercise:

1.Write The Commands To Perform Basic Arithmetic In R.

CODE:

a=10

b=5

print("Addition :")

print(a+b)

print("Subtraction :")

print(a-b)

print("Multiplication :")

print(a\*b)

print("Division :")

print(a/b)

print("Power :")

print(a\*\*b)

Output:  
[1] "Addition :"

[1] 15

[1] "Subtraction :"

[1] 5

[1] "Multiplication :"

[1] 50

[1] "Division :"

[1] 2

[1] "Power :"

[1] 1e+05

2. Display a String on R Console.

CODE:

str="abcdef"

print(str)

Output:

[1] "abcdef"

3. Declare Variables In R And Also Write The Commands For Retrieving The Value Of The Stored Variables In R Console.

CODE:  
# leftward assignment

a <- 1

b = 2

# rightward assignment

4 -> d

5 ->> e

print(a)

print(b)

print(e)

print(d)

Output:  
[1] 1

[1] 2

[1] 5

[1] 4

4. Write R script to calculate the area of Rectangle.

CODE:  
b=as.integer(readline(prompt="Enter breadth of the rectangle :"))

l=as.integer(readline(prompt=" enter the length of the rectangle :"))

print("Area of rectangle :")

print(l\*b)

OUTPUT:

Enter breadth of the rectangle :2

enter the length of the rectangle :4

[1] "Area of rectangle :"

[1] 8

5.Write Commands In R Console To Determine The Type Of Variable

CODE:  
a=1

b="abc"

c=TRUE

d=1+1i

print(typeof(a))

print(typeof(b))

print(typeof(c))

print(typeof(d))

OUTPUT:

[1] "double"

[1] "character"

[1] "logical"

[1] "complex"

6.Enumerate The Process To Check Whether A Given Input Is Numeric , Integer , Double, Complex in R.

CODE:

a=10

if(class(a)=="numeric"){

print("Numeric = TRUE")

}else{

print("Numeric = FALSE")

}

if(class(a)=="character"){

print("Character = TRUE")

}else{

print("Character = FALSE")

}

if(class(a)=="logical"){

print("Logical = TRUE")

}else{

print("Logical = FALSE")

}

if(class(a)=="complex"){

print("Complex = TRUE")

}else{

print("Complex = FALSE")

}

OUTPUT:

[1] "Numeric = TRUE"

[1] "Character = FALSE"

[1] "Logical = FALSE"

[1] "Complex = FALSE"

7. Illustration of Vector Arithmetic.

CODE:

v1=c(1,2,3)

v2=c(1,2,3)

print("Vector Addition :")

print(v1+v2)

print("Vector Subtraction :")

print(v1-v2)

print("Vector Multiplication ")

print(v1\*v2)

print("Vector Division :")

print(v1/v2)

OUTPUT:

[1] "Vector Addition :"

[1] 2 4 6

[1] "Vector Subtraction :"

[1] 0 0 0

[1] "Vector Multiplication "

[1] 1 4 9

[1] "Vector Division :"

[1] 1 1 1

8. Write an R Program to Take Input From User.

Input name as “Jack” and age as 17.

The program should display the output as

“Hai , Jack next year you will be 18 years old”

CODE:

name=readline(prompt = "Enter your name :")

age = as.integer(readline(prompt = "Enter your age :"))

print(paste("Hi, ",name," next year you will be ",age+1," years old"))

OUTPUT:  
Enter your name :jack

Enter your age :18

[1] "Hi, jack next year you will be 19 years old"

**2. DATA STRUCTURES IN R**

Exercise:

1. Perform Matrix Addition & Subtraction in R

CODE:

r1=c(1,2,3,4)

r2=c(9,10,11,12)

mat1=matrix(r1,nrow=2,ncol=2,byrow = TRUE)

row.names(mat1)=c("A","B")

colnames(mat1)=c("C","D")

mat2=matrix(r2,nrow=2,ncol=2,byrow = TRUE)

row.names(mat2)=c("A","B")

colnames(mat2)=c("C","D")

print("First matrix :")

print(mat1)

print("Second matrix :")

print(mat2)

print("Matrix Addition :")

print(mat1+mat2)

print("Matrix subtraction :")

print(mat2-mat1)

OUTPUT:  
[1] "First matrix :"

C D

A 1 2

B 3 4

[1] "Second matrix :"

C D

A 9 10

B 11 12

[1] "Matrix Addition :"

C D

A 10 12

B 14 16

[1] "Matrix subtraction :"

C D

A 8 8

B 8 8

1. Perform Scalar multiplication and matrix multiplication in R

CODE:

r1=c(1,2,3,4)

r2=c(9,10,11,12)

mat1=matrix(r1,nrow=2,ncol=2,byrow = TRUE)

row.names(mat1)=c("A","B")

colnames(mat1)=c("C","D")

mat2=matrix(r2,nrow=2,ncol=2,byrow = TRUE)

row.names(mat2)=c("A","B")

colnames(mat2)=c("C","D")

print("Scalar multiplication :")

print(mat1\*2)

print("Matrix Multiplication :")

print(mat1\*mat2)

OUTPUT:  
[1] "Scalar multiplication :"

C D

A 2 4

B 6 8

[1] "Matrix Multiplication :"

C D

A 9 20

B 33 48

3) Find Transpose of matrix in R.

CODE:

r1=c(1,2,3,4)

r2=c(9,10,11,12)

mat1=matrix(r1,nrow=2,ncol=2,byrow = TRUE)

row.names(mat1)=c("A","B")

colnames(mat1)=c("C","D")

print("Matrix :")

print(mat1)

print("Transpose of Matrix :")

print(t(mat1))

OUTPUT:  
[1] "Matrix :"

C D

A 1 2

B 3 4

[1] "Transpose of Matrix :"

A B

C 1 3

D 2 4

4)Perform the operation of combining matrices in R using cbind() and rbind() functions.

CODE:

r1=c(1,2,3,4)

r2=c(9,10,11,12)

mat1=matrix(r1,nrow=2,ncol=2,byrow = TRUE)

row.names(mat1)=c("A","B")

colnames(mat1)=c("C","D")

mat2=matrix(r2,nrow=2,ncol=2,byrow = TRUE)

row.names(mat2)=c("A","B")

colnames(mat2)=c("C","D")

print("Original matrices :")

print(mat1)

print(mat2)

print("Matrices after Cbind :")

print(cbind(mat1,mat2))

print("Matrices after Rbind :")

print(rbind(mat1,mat2))

OUTPUT:  
[1] "Original matrices :"

C D

A 1 2

B 3 4

C D

A 9 10

B 11 12

[1] "Matrices after Cbind :"

C D C D

A 1 2 9 10

B 3 4 11 12

[1] "Matrices after Rbind :"

C D

A 1 2

B 3 4

A 9 10

B 11 12

5)Deconstruct a matrix in R

CODE:  
library(reshape)

r1=c(1,2,3,4)

r2=c(9,10,11,12)

mat1=matrix(r1,nrow=2)

r1=c(1,2,3,4)

r2=c(9,10,11,12)

print("Matrix :")

print(mat1)

print("Matrix after deconstruction :")

print(melt.matrix(mat1))

OUTPUT:

[1] "Matrix :"

[,1] [,2]

[1,] 1 3

[2,] 2 4

[1] "Matrix after deconstruction :"

X1 X2 value

1 1 1 1

2 2 1 2

3 1 2 3

1. 2 2 4

6)Perform array manipulation in R .

CODE:

arr=array(data=c(1,2,3,4),dim=4,dimnames = row.names("a"))

print(arr)

print("Sum of Array :")

print(sum(arr))

print("adding an element into the array")

arr=array(data=c(arr[0:2],5,3:4))

print(arr)

OUTPUT:  
[1] 1 2 3 4

[1] "Sum of Array :"

[1] 10

[1] "adding an element into the array"

[1] 1 2 5 3 4

7)Perform calculations across array elements in an array using the apply() function.

CODE:  
print("Array :")

arr=array(data=c(1,2,3,4),dim=c(2,2),dimnames = row.names("a"))

print(arr)

print("Applying sum into the array :")

print(apply(arr,1,sum))

OUTPUT:  
[1] "Array :"

[,1] [,2]

[1,] 1 3

[2,] 2 4

[1] "Applying sum into the array :"

[1] 4 6

8)Demonstrate Factor data structure in R.

CODE:

fact=factor(c("apple", "banana","orange","apple","pear","banana","strawberry"))

print(fact)

OUTPUT:

[1] apple banana orange apple pear banana strawberry

Levels: apple banana orange pear strawberry

9)Create a data frame and print the structure of the data frame in R.

CODE:

name=c("ab","cd","ef","gh")

age=c(10,20,30,40)

gender=c("M","F","M","F")

df=data.frame(name,age,gender)

print("Dataframe :")

print(df)

print("Stucture of Dataframe :")

print(str(df))

OUTPUT:  
[1] "Dataframe :"

name age gender

1 ab 10 M

2 cd 20 F

3 ef 30 M

4 gh 40 F

[1] "Stucture of Dataframe :"

'data.frame': 4 obs. of 3 variables:

$ name : chr "ab" "cd" "ef" "gh"

$ age : num 10 20 30 40

$ gender: chr "M" "F" "M" "F"

NULL

10)Demonstrate the creation of S3 class in R.

CODE:

x <- list(name ="Abc", reg\_no = 10)

class(x)<-"student"

print(x)

OUTPUT:

$name

[1] "Abc"

$reg\_no

[1] 10

attr(,"class")

[1] "student"

11) Demonstrate the creation of S4 class in R.

CODE:

setClass("Student",slots = list(name="character",reg\_no="numeric"))

student1=new("Student",name="abc",reg\_no=10)

print(student1)

OUTPUT:  
An object of class "Student"

Slot "name":

[1] "abc"

Slot "reg\_no":

[1] 10

12) Demonstrate the creation of Reference class in R by defining a class called students with fields – Name, Age , GPA. Also illustrate how the fields of the object can be accessed using the $ operator. Modify the Name field by reassigning the name to Paul.

CODE:

student=setRefClass("student",fields = list(name="character",age="numeric",GPA="numeric"))

student1=student(name="abc",age=18,GPA=9)

print("Class :")

print(student1)

print("Accessing with $ operator :")

print(student1$name)

print("Modifying name field :")

student1$name="paul"

print(student1)

OUTPUT:  
[1] "Class :"

Reference class object of class "student"

Field "name":

[1] "abc"

Field "age":

[1] 18

Field "GPA":

[1] 9

[1] "Accessing with $ operator :"

[1] "abc"

[1] "Modifying name field :"

Reference class object of class "student"

Field "name":

[1] "paul"

Field "age":

[1] 18

Field "GPA":

[1] 9

**3. WORKING WITH LOOPING & FUNCTION IN R**

Excercies:

1.Write a program to check whether an integer (entered by the user) is a prime number or not using control statements.

CODE:  
n=as.integer(readline(prompt = "Enter a number :"))

start=2

flag=0

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

if(n%%i==0){

flag=1

}

}

if(flag==0){

print("is a prime number")

}else if(flag==1){

print("is not a prime number")

}

OUTPUT:  
Enter a number :7

[1] "is a prime number"

Enter a number :6

[1] "is not a prime number"

2.Write a program to check whether a number entered by the user is positive number or a negative number or zero.

CODE:  
n=as.integer(readline(prompt = "Enter a number :"))

if(n==0){

print("Zero")

}else if(n>0){

print("Positive number")

}else if(n<0){

print("Negetive number")

}

OUTPUT:  
Enter a number :0

[1] "Zero"

Enter a number :1

[1] "Positive number"

Enter a number :-1

[1] "Negetive number"

3.Write a program to check whether a number is an Armstrong number or not using a while loop.

CODE:

n=as.integer(readline(prompt ="Enter a number :"))

s=0

c=n

while(n>0){

x=n%%10

s=s+x\*x\*x

n=n%/%10

}

if(s==c)

{

print("is Armstrong ")

}else{

print("is not Armstrong")

}

OUTPUT:  
Enter a number :153

[1] "is Armstrong "

Enter a number :169

[1] "is not Armstrong"

4.Write a program to demonstrate Repeat Loop in R

CODE:

i <- 1

repeat {

print(i)

i <- i + 1

if(i >5) {

break

}

}

CODE:  
[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

5.Using functions develop a simple calculator in R.

CODE:

summ<-function(x,y){

print(x+y)

}

sub<-function(x,y){

print(x-y)

}

mul<-function(x,y){

print(x\*y)

}

div<-function(x,y){

print(x/y)

}

power<-function(x,y){

print(x\*\*y)

}

choice<-function(){

cat("1=addition\n2=subraction\n3=multiplication\n4=division\n5=power")

ch=as.integer(readline(prompt = "Choose an operation :"))

return(ch)

}

x=choice()

if(x==1){

x=as.integer(readline(prompt = "enter left hand side operend :"))

y=as.integer(readline(prompt = "enter right hand side operend :"))

summ(x,y)

}else if(x==2){

x=as.integer(readline(prompt = "enter left hand side operend :"))

y=as.integer(readline(prompt = "enter right hand side operend :"))

sub(x,y)

}else if(x==3){

x=as.integer(readline(prompt = "enter left hand side operend :"))

y=as.integer(readline(prompt = "enter right hand side operend :"))

mul(x,y)

}else if(x==4){

x=as.integer(readline(prompt = "enter left hand side operend :"))

y=as.integer(readline(prompt = "enter right hand side operend :"))

div(x,y)

}else if(x==5){

x=as.integer(readline(prompt = "enter left hand side operend :"))

y=as.integer(readline(prompt = "enter right hand side operend :"))

power(x,y)

}else{

print("Invalid input... Try again")

choice()

}

OUTPUT:

1=addition

2=subraction

3=multiplication

4=division

5=power

Choose an operation :4

enter left hand side operend :10

enter right hand side operend :2

[1] 5

6. Demonstrate the creation of a complex number in R.

CODE:

a <- 1

b <- 1

x <- complex(real = a, imaginary = b)

print(x)

OUTPUT:

[1] 1+1i

7.Write a program to multiply two numbers using a function with a default value. Assume default value as NULL.

CODE:

mul<-function(x,y){

return(x\*y)

}

x=as.integer(readline(prompt = "enter a number :"))

y=as.integer(readline(prompt = "enter another number :"))

print(mul(x,y))

OUTPUT:

enter a number :2

enter another number :3

[1] 6

8.Find sum, mean and product of vector elements using built-in functions.

CODE:

v=c(1:10)

print(paste("Sum of vector :",sum(v[1:10])))

print(paste("Mean :",mean(v)))

print(paste("Product :",prod(v)))

OUTPUT:

[1] "Sum of vector : 55"

[1] "Mean : 5.5"

[1] "Product : 3628800"

9.Sort a vector in R using sort() function. Also find the index of the sorted vector.

CODE:

v=c(1,4,5,6,2,3,7)

v=sort(v)

print("Vector after sorting :")

print(v)

OUTPUT:

[1] "Vector after sorting :"

[1] 1 2 3 4 5 6 7

10.Find the L.C.M of two numbers entered by the user by creating a user-defined function.

CODE:

lcm <- function(x, y) {

v1=c()

v2=c()

for(i in 1:10){

v1=append(v1,i\*x)

}

for(i in 1:10){

v2=append(v2,i\*y)

}

for(i in v1){

for(j in v2){

if(i==j){

return(i)

}

}

}

}

num1 = as.integer(readline(prompt = "Enter first number: "))

num2 = as.integer(readline(prompt = "Enter second number: "))

print(paste("The LCM is", lcm(num1, num2)))

OUTPUT:

Enter first number: 3

Enter second number: 15

[1] "The LCM is 15"

**4. IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY & RECURSION**

Exercises:

1)Demonstrate Vector Recycling in R.

CODE:

v1=c(1,2,3,4,5)

v2=c(1,2)

print(v1+v2)

OUTPUT:

[1] 2 4 4 6 6

2. Demonstrate the usage of apply function in R

CODE:

print("matrix :")

r=c(1,2,3,4)

mat=matrix(r1,ncol = 2)

print(mat)

print("Applying sum into the columns of matrix :")

print(apply(arr,2,sum))

OUTPUT:

[1] "matrix :"

[,1] [,2]

[1,] 1 3

[2,] 2 4

[1] "Applying sum into the columns of matrix :"

[1] 3 7

3. Demonstrate the usage of lapply function in R

CODE:

data <- data.frame(a = c(1, 2, 3),b = c(1, 2, 3),c = c(1, 2, 3))

print(data)

print(lapply(data,sum))

OUTPUT:

a b c

1 1 1 1

2 2 2 2

3 3 3 3

$a

[1] 6

$b

[1] 6

$c

[1] 6

4. Demonstrate the usage of sapply function in R

CODE:  
data <- data.frame(a = c(1, 2, 3),b = c(1, 2, 3),c = c(1, 2, 3))

print(data)

print(sapply(data,sum))

OUTPUT:  
 a b c

1 1 1 1

2 2 2 2

3 3 3 3

a b c

6 6 6

5. Demonstrate the usage of tapply function in R

CODE:  
data <- data.frame(a = c(3, 4, 1),b = c(6, 2, 3),c = c(5,8,10))

print(data)

print(tapply(data$c,data$a,mean))

OUTPUT:  
a b c

1 3 6 5

2 4 2 8

3 1 3 10

1 3 4

10 5 8

6. Demonstrate the usage of mapply function in R

CODE:

v1 <- c(1, 2, 3, 4, 5)

v2 <- c(2, 4, 1, 2, 6)

print(mapply(max, v1, v2))

OUTPUT:  
[1] 2 4 3 4 6

7. Sum of Natural Numbers using Recursion

CODE:

n=as.integer(readline(prompt = "Enter number of natural numbers :"))

sum=0

for(i in 1:n){

sum=sum+i

}

print(paste("Sum of ",n," natural numbers :",sum))

OUTPUT:

[1] "Sum of 10 natural numbers : 55"

8. Write a program to generate Fibonacci sequence using Recursion in R

CODE:

fibo<- function(n) {

if(n<= 1) {

return(n)

} else {

return(fibo(n-1) + fibo(n-2))

}

}

n = as.integer(readline(prompt = "Enter number of digits in series :"))

print("Fibonacci sequence:")

for(i in 0:(n-1)) {

print(fibo(i))

}

OUTPUT:

[1] "Fibonacci sequence:"

[1] 0

[1] 1

[1] 1

[1] 2

[1] 3

[1] 5

9. Write a program to find factorial of a number in R using recursion.

CODE:

fact<- function(n) {

if(n <= 1) {

return(1)

} else {

return(n\*fact(n-1))

}

}

x=as.integer(readline(prompt = "Enter a number :"))

print(paste("factorial :",fact(x)))

OUTPUT:

Enter a number :4

[1] "factorial : 24"