LAB EXERCISES

ITA0443-STATISTICS WITH R-PROGRAMMING

NAME: N M S N V ARUN

REG.NO:192110529

1.BASIC OPERATIONS IN R

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

programm:

```
5 + 10
```

[1]15

```
5 - 10
```

[1]-5

```
5 * 10
```

[1]50

```
10 / 5
```

[1]2

```
10 %% 3
```

[1]1

```
2 ^ 3
```

[1]8

2. Display a String on R Console.

programm:

```
print("Hello World!")
cat("Hello World!")
```

OUT PUT

[1] "HELLO WORLD"

HELLO WORLD

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

The Stored Variables In R Console.

programm:

```
x <- 5
y <- 10
```

```
z <- "Hello World!"
x
[1]5
y
[1]10
z
[1]"Hello World!"
```

```
x <- 5
y <- 10
z <- "Hello World!"
print(x)
[1] 5
print(y)
[1] 10
print(z)
[1] "Hello World!"
```

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

programm:

```
length <- as.numeric(readline(prompt="Enter the length of the rectangle: "))
width <- as.numeric(readline(prompt="Enter the width of the rectangle: "))
area <- length * width
print(paste("The area of the rectangle is", area))
```

OUT PUT

```
Enter the length of the rectangle: 5
Enter the width of the rectangle: 10
[1] "The area of the rectangle is 50"
```

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

programm:

```
x <- 5
class(x)
```

[1]"numeric"

y <- "Hello World!"

class(y)

[1]"character"

x <- 5

typeof(x)

[1]"double"

y <- "Hello World!"

typeof(y)

[1]"character"

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

programm:

x <- 5

is.numeric(x)

[1]TRUE

is.integer(x)

[1]TRUE

is.double(x)

[1]FALSE

is.complex(x)

[1]FALSE

7. Illustration of Vector Arithmetic.

programm:

x <- c(1, 2, 3)

y <- c(4, 5, 6)

z <- x + y

```
z

# [1] 5 7 9

x <- c(1, 2, 3)

y <- c(4, 5, 6)

z <- x - y

z

# [-3, -3, -3]

x <- c(1, 2, 3)

y <- c(4, 5, 6)

z <- x * y

z

# [4, 10, 18]

x <- c(1, 2, 3)

y <- c(4, 5, 6)

z <- x / y

z

# [0.25, 0.4, 0.5]
```

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"

programm:

```
name <- readline(prompt="Enter your name: ")

age <- as.numeric(readline(prompt="Enter your age: "))

message <- paste("Hai, ", name, " next year you will be ", age + 1, " years old")

print(message)
```

OUTPUT

```
Enter your name: Jack

Enter your age: 17
```

[1] "Hai, Jack next year you will be 18 years old"

2.DATA STRUCTURES IN R

1) Perform Matrix Addition & Subtraction in R

programm:

```
A <- matrix(1:4, nrow = 2, ncol = 2)

B <- matrix(5:8, nrow = 2, ncol = 2)

C <- A + B

print(C)

D <- A - B

print(D)
```

OUTPUT:

```
     [,1] [,2]

[1,]    6    8

[2,]   10   12


[,1] [,2]

[1,] -4 -4

[2,] -4 -4
```

2) Perform Scalar multiplication and matrix multiplication in R

programm:

```
A <- matrix(1:4, nrow = 2, ncol = 2)

B <- 2 * A

print(B)

C <- A %*% t(A)

print(C)
```

OUTPUT

```
     [,1] [,2]

[1,]    2    4

[2,]    6    8
```

```
     [,1] [,2]
[1,]   10   14
[2,]   14   20
```

3) Find Transpose of matrix in R.

programm:

```
A <- matrix(1:4, nrow = 2, ncol = 2)
B <- t(A)
print(B)
```

OUTPUT:

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

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

functions.

programm:

```
A <- matrix(1:4, nrow = 2, ncol = 2)
B <- matrix(5:8, nrow = 2, ncol = 2)
C <- cbind(A, B)
print(C)
D <- rbind(A, B)
print(D)
```

OUTPUT:

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    5    6
[2,]    3    4    7    8


     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

[4,] 7 8

5) Deconstruct a matrix in R

programm:

```
A <- matrix(1:4, nrow = 2, ncol = 2)

a1 <- A[1,1]

a2 <- A[1,2]

a3 <- A[2,1]

a4 <- A[2,2]

print(a1)

print(a2)

print(a3)

print(a4)
```

OUTPUT:

```
[1] 1

[1] 2

[1] 3

[1] 4
```

6) Perform array manipulation in R

programm:

```
x <- c(1, 2, 3, 4)

y <- matrix(rep(x, times = 2), ncol = 2, byrow = TRUE)

z <- array(1:24, dim = c(2, 3, 4))

print(x)

print(y)

print(z)
```

OUTPUT:

```
[1] 1 2 3 4

     [,1] [,2]

[1,]    1    1

[2,]    2    2

[3,]    3    3
```

```
[4,]    4    4

     [,1] [,2] [,3]

[1,]    1    3    5

[2,]    2    4    6

     [,1] [,2] [,3]

[1,]    7    9   11
```

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

programm:

```
x <- matrix(1:6, nrow = 2, ncol = 3)

col_sums <- apply(x, 2, sum)

print(col_sums)

row_means <- apply(x, 1, mean)

print(row_means)
```

OUTPUT:

```
[1]  3  5  7

[1] 2.5 3.5
```

8) Demonstrate Factor data structure in R.

programm:

```
x <- c("apple", "banana", "cherry", "banana", "apple")

x_factor <- factor(x)

print(x_factor)
```

OUTPUT:

```
[1] apple  banana cherry  banana apple

Levels: apple banana cherry
```

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

programm:

```
df <- data.frame(Name = c("SHASHI", "TAKESH", "SAI"),

                 Age = c(19, 20, 21),

                 Gender = c("Male", "male", "Male"))

str(df)
```

OUTPUT:

```
$ Name  : Factor w/ 3 levels "SHASHI","TAKESH","SAI": 3 1 2
 $ Age   : num  19 20 21
 $ Gender: Factor w/ 2 levels "male","Male": 2 1 2
```

10) Demonstrate the creation of S3 class in R.

programm:

```
Person <- function(name, age) {
  structure(list(name = name, age = age), class = "Person")
}



print.Person <- function(person) {
  cat(paste("Name:", person$name, "\nAge:", person$age, "\n"))
}
p1 <- Person("John", 30)



print(p1)
```

OUTPUT:

```
Name: John
Age: 30
```

11) Demonstrate the creation of S4 class in R.

programm:

```
setClass("Person", representation(name = "character", age = "numeric"))
setMethod("print", "Person", function(object) {
  cat(paste("Name:", object@name, "\nAge:", object@age, "\n"))
})
p1 <- new("Person", name = "John", age = 30)
print(p1)
```

OUTPUT:

Name: John

Age: 30

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.

programm:

```
library(methods)
students <- setRefClass("students",
  fields = list(
    Name = "character",
    Age = "numeric",
    GPA = "numeric"
  )
)
s1 <- students$new(Name = "John", Age = 25, GPA = 3.5)
cat("Name:", s1$Name, "\nAge:", s1$Age, "\nGPA:", s1$GPA, "\n")
s1$Name <- "Paul"
cat("Name:", s1$Name, "\nAge:", s1$Age, "\nGPA:", s1$GPA, "\n")
```

OUTPUT:

Name: John

Age: 25

GPA: 3.5

Name: Paul

Age: 25

GPA: 3.5

3.WORKING WITH LOOPING AND FUNCTIONS IN R

1.Write a program to check whether an integer (entered by the user) is a prime number

or not using control statements.

programm:

```
num <- as.integer(readline(prompt="Enter an integer: "))
flag <- 1
if(num == 2) {
  flag <- 0
} else {
  for(i in 2:(num-1)) {
    if((num %% i) == 0) {
      flag <- 0
      break
    }
  }
}
if(flag == 0) {
  cat("The entered number is not a prime number.")
} else {
  cat("The entered number is a prime number.")
}
```

OUTPUT:

```
ENTER AN INTEGER: 7
[1]The entered number is prime number
```

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

programm:

```
num <- as.integer(readline(prompt="Enter a number: "))
if(num > 0) {
  cat("The entered number is a positive number.")
} else if(num < 0) {
  cat("The entered number is a negative number.")
```

```
} else {
  cat("The entered number is zero.")
}
```

OUTPUT:

```
Enter a number:9
[1] The entered number is a positive number
```

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

programm:

```
num <- as.integer(readline(prompt="Enter a number: "))
digits <- nchar(as.character(num))
sum_cubes <- 0
temp_num <- num
while(temp_num > 0) {
  digit <- temp_num %% 10
  sum_cubes <- sum_cubes + (digit^digits)
  temp_num <- floor(temp_num / 10)
}
if(sum_cubes == num) {
  cat("The entered number is an Armstrong number.")
} else {
  cat("The entered number is not an Armstrong number.")
}
```

OUTPUT:

```
Enter a number:153
[1] entered number is an Armstrong number
```

4.Write a program to demonstrate Repeat Loop in R

programm:

```
count <- 1
```

```
repeat{
  print(count)
  count <- count + 1
  if (count > 5) {
    break
  }
}
```

OUTPUT:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

5.Using functions develop a simple calculator in R.

programm:

```
calculate <- function(num1, num2, operator) {
  if (operator == "+") {
    return(num1 + num2)
  } else if (operator == "-") {
    return(num1 - num2)
  } else if (operator == "*") {
    return(num1 * num2)
  } else if (operator == "/") {
    return(num1 / num2)
  } else {
    return("Invalid operator")
  }
}

result <- calculate(5, 3, "+")
```

```
print(result)


result <- calculate(5, 3, "-")

print(result)


result <- calculate(5, 3, "*")

print(result)


result <- calculate(5, 3, "/")

print(result)


result <- calculate(5, 3, "^")

print(result)
```

OUTPUT:

```
[1] 8

[1] 2

[1] 15

[1] 1.666667

[1] "Invalid operator"
```

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

programm:

```
z1 <- complex(real = 1, imaginary = 2)

print(z1)

z2 <- 3 + 4i

print(z2)
```

OUTPUT:

```
[1] 1+2i

[1] 3+4i
```

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

Assume default value as NULL.

programm:

```
multiply <- function(x, y = NULL) {
  if (is.null(y)) {
    y <- 1
  }
  return (x * y)
}


result <- multiply(5)
print(result)


result <- multiply(5, 3)
print(result)
```

OUTPUT:

```
[1] 5
[1] 15
```

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

programm:

```
vec <- c(1, 2, 3, 4, 5)
sum_of_elements <- sum(vec)
print(sum_of_elements)
mean_of_elements <- mean(vec)
print(mean_of_elements)
product_of_elements <- prod(vec)
print(product_of_elements)
```

OUTPUT:

```
[1] 15
```

[1] 3

[1] 120

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

programm:

```
vec <- c(5, 3, 2, 4, 1)

sorted_vec <- sort(vec)

print(sorted_vec)

index_sorted_vec <- order(vec)

print(index_sorted_vec)
```

OUTPUT:

[1] 1 2 3 4 5

[1] 5 4 3 2 1

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

programm:

```
find_lcm <- function(x, y) {

  return (x * y / gcd(x, y))

}

x <- as.integer(readline(prompt = "Enter the first number: "))

y <- as.integer(readline(prompt = "Enter the second number: "))

lcm <- find_lcm(x, y)

print(paste("The LCM of", x, "and", y, "is", lcm))
```

OUTPUT:

[1]Enter the first number: 12

[1]Enter the second number:4

[1]The LCM of 12 and 4 is 12