Name:Vishwa

Regno:192124161

EXERCISE:1

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

```
  vec1 <- c(0, 2)

> vec2 <- c(2, 3)

>

> cat ("Addition of vectors :", vec1 + vec2, "\n")

Addition of vectors : 2 5

> cat ("Subtraction of vectors :", vec1 - vec2, "\n")

Subtraction of vectors : -2 -1

> cat ("Multiplication of vectors :", vec1 * vec2, "\n")

Multiplication of vectors : 0 6

> cat ("Division of vectors :", vec1 / vec2, "\n")

Division of vectors : 0 0.6666667

> cat ("Modulo of vectors :", vec1 %% vec2, "\n")

Modulo of vectors : 0 2

> cat ("Power operator :", vec1 ^ vec2)

Power operator : 0 8>
```

2. Display a String on R Console.

```
s <-readline(prompt("enter the string"))

print(s)
```

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

The Stored Variables In R Console.

```
s <-readline(prompt("enter the name:"))

print(s)
```

4. Write R script to calculate the area of Rectangle

```
l<-as.integer(readline("enter the length of rectangle:"))

b<-as.integer(readline("enter the breadth of rectangle:"))

print(l*b)

enter the length of rectangle:2

enter the breadth of rectangle:5

10
```

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

```
#define variable x

x <- c("Andy", "Bob", "Chad", "Dave", "Eric", "Frank")

class(x)

[1] "character"
```

Its defines that x is a character variable.

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

```
is.integer();

is.complex();

is.double();

is.complex();
```

7. Illustration of Vector Arithmetic.

```
>   a = c(1, 3, 5, 7)

>   b = c(1, 3, 4, 8)

> a-b

[1]   0   0   1 -1

> a*b

[1]   1   9 20 56
```

```
> 5 * a
[1]   5 15 25 35
```

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

```
> my.name <- readline(prompt="enter name:")
enter name:hari
> my.age <- readline(prompt="enter age:")
enter age:17
> my.age <- as.integer(my.age)
> print(paste("Hi,",my.name,"next year u will be",my.age+1,"years old."))
[1] "Hi, hari next year u will be 18 years old."
```

Exercise: 2

1) Perform Matrix Addition & Subtraction in R

```
> matrix1 <- matrix(1:6, nrow = 2)
> matrix2 <- matrix(7:12, nrow = 2)
> matrix_sum <- matrix1 + matrix2
> print(matrix_sum)
     [,1] [,2] [,3]
[1,]    8   12   16
[2,]   10   14   18
> matrix_diff <- matrix1 - matrix2
> print(matrix_diff)
     [,1] [,2] [,3]
[1,]   -6   -6   -6
[2,]   -6   -6   -6
>
```

2) Perform Scalar multiplication and matrix multiplication in R

```
> matrix1 <- matrix(1:6, nrow = 2)

> scalar_mult <- 2 * matrix1

> print(scalar_mult)

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

[1,]      2      6     10

[2,]      4      8     12

> matrix2 <- matrix(c(1, 2, 3, 4), nrow = 2)

> matrix_prod <- matrix1 %*% matrix2

> print(matrix_prod)

        [,1] [,2]

[1,]    22     28

[2,]    49     64
```

3) Find Transpose of matrix in R.

```
matrix1 <- matrix(1:6, nrow = 2)

matrix_transpose <- t(matrix1)

print(matrix_transpose)

        [,1] [,2]

[1,]      1      2

[2,]      3      4

[3,]      5      6

>
```

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

CBIND:

```
matrix1 <- matrix(1:4, nrow = 2)

matrix2 <- matrix(5:8, nrow = 2)

matrix_combined <- cbind(matrix1, matrix2)
```

```
print(matrix_combined)

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

RBIND:

```
matrix1 <- matrix(1:4, nrow = 2)

matrix2 <- matrix(5:8, nrow = 2)

 matrix_combined <- rbind(matrix1, matrix2)

print(matrix_combined)

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

5) Deconstruct a matrix in R

```
matrix1 <- matrix(1:6, nrow = 2)

element1 <- matrix1[1, 1]

element2 <- matrix1[2, 2]

print(element1)

print(element2)

row1 <- matrix1[1, ]

row2 <- matrix1[2, ]

print(row1)

print(row2)

column1 <- matrix1[, 1]

column2 <- matrix1[, 2]

print(column1)

print(column2)
```

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

6) Perform array manipulation in R .

```
my_array <- array(1:24, dim = c(2, 3, 4))
print(my_array)
, , 1
   [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
, , 2
    [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
, , 3
    [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18
, , 4
     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

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

```
 my_array <- c(1, 2, 3, 4, 5)
> my_calculation <- function(x) {
```

```
+     return(x * 2)

+ }

print(new_array)
```

output:

```
[1]   2   4   6   8 10
```

8) Demonstrate Factor data structure in R.

```
my_data <- c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female",
"Male")

my_factor <- factor(my_data)

print(my_factor)

print(levels(my_factor))

print(table(my_factor))

[1] Male     Female Male     Male     Female Female Male     Female Male

Levels: Female Male

my_factor

Female      Male

       4          5
```

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

```
my_data <- data.frame(

  name = c("Alice", "Bob", "Charlie", "David"),

  age = c(25, 30, 35, 40),

  salary = c(50000, 60000, 70000, 80000)

)

# Print the data frame

print(my_data)

str(my_data)

        name age salary

1      Alice   25   50000
```

```
2          Bob    30    60000
3   Charlie    35    70000
4       David    40    80000
```

10) Demonstrate the creation of S3 class in R.

```
my_class <- function(x) {
   obj <- list(data = x)
   class(obj) <- "my_class"
   return(obj)
}
my_obj <- my_class(1:10)
print(my_obj)
print(class(my_obj))
```

```
$data
 [1]   1    2    3    4    5    6    7    8    9 10

attr(,"class")
[1] "my_class"
```

11) Demonstrate the creation of S4 class in R.

```
setClass(
   "my_class",
   slots = list(
      data = "numeric",
      name = "character"
   )
)
my_obj <- new("my_class", data = 1:10, name = "My Data")
print(my_obj)
print(slot(my_obj, "data"))
print(slot(my_obj, "name"))
```

Output:

```
An object of class "my_class"

Slot "data":

 [1]  1  2  3  4  5  6  7  8  9 10


Slot "name":

[1] "My Data"
```

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.

```
students <- setRefClass("students",

  fields = list(

    Name = "character",

    Age = "numeric",

    GPA = "numeric"

  )

)

my_student <- students(Name = "John", Age = 20, GPA = 3.5)

print(my_student$Name)

print(my_student$Age)

print(my_student$GPA)

my_student$Name <- "Paul"

print(my_student$Name)
```

The output of the above code will be:

csharp

Copy code

```
[1] "John"

[1] 20

[1] 3.5

[1] "Paul"
```