#Q.4]Implement basic functions and commands in R Programming.

# A simple R function to check

# whether x is even or odd

evenOdd = function(x){

if(x %% 2 == 0)

return("even")

else

return("odd")

}

print(evenOdd(4))

print(evenOdd(3))

# A simple R function to calculate

# area of a circle

areaOfCircle = function(radius){

area = pi\*radius^2

return(area)

}

print(areaOfCircle(2))

# A simple R function to calculate

# area and perimeter of a rectangle

Rectangle = function(length, width){

area = length \* width

perimeter = 2 \* (length + width)

# create an object called result which is

# a list of area and perimeter

result = list("Area" = area, "Perimeter" = perimeter)

return(result)

}

resultList = Rectangle(2, 3)

print(resultList["Area"])

print(resultList["Perimeter"])

# A simple R program to demonstrate

# passing arguments to a function

Rectangle = function(length=5, width=4){

area = length \* width

return(area)

}

# Case 1:

print(Rectangle(2, 3))

# Case 2:

print(Rectangle(width = 8, length = 4))

# Case 3:

print(Rectangle())

#Q.6]Write a program using R programming for Fibonacci series.

print\_fibonacci <- function(n) {

a <- 0

b <- 1

cat("Fibonacci Sequence:")

for (i in 1:n) {

cat(a, " ")

next\_num <- a + b

a <- b

b <- next\_num

}

}

# Example usage

number\_of\_terms <- 10

print\_fibonacci(number\_of\_terms)

#7.Write a program using R programming to find Armstrong number.

# Function to check for an Armstrong number

is\_armstrong\_number <- function(number) {

num <- number

num\_of\_digits <- nchar(num)

sum\_of\_digits <- 0

while (num > 0) {

digit <- num %% 10

sum\_of\_digits <- sum\_of\_digits + digit^num\_of\_digits

num <- num %/% 10

}

return(sum\_of\_digits == number)

}

# Example usage

number\_to\_check <- 153

if (is\_armstrong\_number(number\_to\_check)) {

cat(number\_to\_check, "is an Armstrong number.")

} else {

cat(number\_to\_check, "is not an Armstrong number.")

}

#2

checkArmstrong <- function(num) {

sum\_of\_digits <- sum(as.numeric(strsplit(as.character(num), "")[[1]]) ^ length(as.character(num)))

is\_armstrong <- sum\_of\_digits == num

if (is\_armstrong) {

return(paste(num, "is an Armstrong number"))

} else {

return(paste(num, "is not an Armstrong number"))

}

}

# Example usage

number <- 371

result <- checkArmstrong(number)

print(result)

#8.Write a program in R programming for prime numbers using.

isPrime <- function(num) {

if (num <= 1) {

return(FALSE)

}

for (i in 2:(num-1)){

if (num %% i == 0) {

return(FALSE)

}

}

return(TRUE)

}

# Example usage

number <- 17

if (isPrime(number)) {

print(paste(number, "is a prime number"))

} else {

print(paste(number, "is not a prime number"))

}

#9.Write a program in R programming to implement sum of series of number

sumOfSeries <- function(n) {

# Initialize sum

total <- 0

# Loop to calculate the sum of numbers from 1 to n

for (i in 1:n) {

total <- total + i

}

return(total)

}

# Example usage

n <- 10 # Change this value to the desired number in the series

result <- sumOfSeries(n)

print(paste("The sum of numbers from 1 to", n, "is:", result))

#10.Write a program in R programming for the implementation of vector

# Creating a numeric vector

numeric\_vector <- c(1, 2, 3, 4, 5)

print("Numeric Vector:")

print(numeric\_vector)

# Accessing elements of the vector

print("\nAccessing Elements:")

print("First element:", numeric\_vector[1])

print("Third element:", numeric\_vector[3])

# Vector arithmetic operations

print("\nVector Arithmetic Operations:")

doubled\_vector <- numeric\_vector \* 2

print("Doubled Vector:", doubled\_vector)

# Adding two vectors element-wise

another\_vector <- c(5, 4, 3, 2, 1)

sum\_vector <- numeric\_vector + another\_vector

print("Sum of Vectors:", sum\_vector)

# Vector concatenation

concatenated\_vector <- c(numeric\_vector, another\_vector)

print("\nConcatenated Vector:")

print(concatenated\_vector)

#11.Write a program in R programming for the implementation of Data frames

# Creating a data frame

student\_data <- data.frame(

Name = c("Alice", "Bob", "Charlie", "David", "Emma"),

Age = c(22, 21, 23, 20, 24),

Grade = c("A", "B", "A", "C", "B"),

stringsAsFactors = FALSE # Ensures character columns are not converted to factors

)

# Displaying the data frame

print("Student Data Frame:")

print(student\_data)

# Accessing columns of the data frame

print("\nAccessing Columns:")

print("Names:")

print(student\_data$Name)

print("Ages:")

print( student\_data$Age)

print("Grades:")

print(student\_data$Grade)

# Adding a new column to the data frame

student\_data$City <- c("New York", "San Francisco", "Los Angeles", "Chicago", "Boston")

print("\nData Frame with Added Column 'City':")

print(student\_data)

# Filtering the data frame based on a condition

selected\_students <- student\_data[student\_data$Age > 21, ]

print("\nStudents older than 21:")

print(selected\_students)

#12.Write a program in R programming for the implementation of Matrices

# Creating a matrix

matrix\_data <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)

print("Matrix:")

print(matrix\_data)

# Accessing elements of the matrix

print("\nAccessing Elements:")

print("Element at row 2, column 3:", matrix\_data[2, 3])

# Basic matrix operations

print("\nBasic Matrix Operations:")

doubled\_matrix <- matrix\_data \* 2

print("Doubled Matrix:")

print(doubled\_matrix)

# Transpose of a matrix

transpose\_matrix <- t(matrix\_data)

print("\nTransposed Matrix:")

print(transpose\_matrix)

# Matrix multiplication

another\_matrix <- matrix(1:8, nrow = 4)

matrix\_product <- matrix\_data %\*% another\_matrix

print("\nMatrix Product:")

print(matrix\_product)

#13.Write a program in R programming for the implementation of list

# Creating a list

my\_list <- list(

Name = "John",

Age = 30,

Grades = c(90, 85, 95),

Passed = TRUE

)

# Displaying the list

print("My List:")

print(my\_list)

# Accessing elements of the list

print("\nAccessing Elements:")

print("Name:" )

print(my\_list$Name)

print("Age:")

print( my\_list$Age)

print("Grades:")

print(my\_list$Grades)

print("Passed:")

print(my\_list$Passed)

# Adding a new element to the list

my\_list$City <- "New York"

print("\nList with Added Element 'City':")

print(my\_list)

# Modifying an element in the list

my\_list$Age <- 32

print("\nModified List (Updated Age):")

print(my\_list)

#15.Find even or odd number using functions in R Programming

# Function to check if a number is even or odd

checkEvenOdd <- function(num) {

if (num %% 2 == 0) {

return(paste(num, "is an even number"))

} else {

return(paste(num, "is an odd number"))

}

}

# Example usage

number <- 7 # Change this number to check for different values

result <- checkEvenOdd(number)

print(result)

#16.Perform String operations using R programming.

#Creating Strings:

# Using double quotes

string1 <- "Hello, World!"

# Using single quotes

string2 <- 'Welcome to R programming!'

#String Concatenation:

# Using paste function

combined\_string <- paste(string1, string2)

print(combined\_string)

#Substring Extraction:

# Substring using substr function

substring <- substr(string1, start = 1, stop = 5)

print(substring)

#String Length:

# Using nchar function

length\_string1 <- nchar(string1)

print(length\_string1)

#Changing Case:

# Convert to uppercase

uppercase\_string1 <- toupper(string1)

print(uppercase\_string1)

# Convert to lowercase

lowercase\_string2 <- tolower(string2)

print(lowercase\_string2)

#Searching in Strings:

# Using grep function

search\_result <- grep("World", string1)

if (length(search\_result) > 0) {

print("Found 'World' in the string")

} else {

print("Not found")

}

#String Splitting:

# Using strsplit function

split\_string <- strsplit(string1, ",")[[1]]

print(split\_string)