**R - LAB PROGRAMS**

**1)** **PERFORM ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION.**

**Aim:** Perform addition, subtraction, multiplication, division, and exponentiation with two numbers (e.g., 8 and 2).

**Code:**

a <- 8

b <- 2

addition <- a + b

subtraction <- a - b

multiplication <- a \* b

division <- a / b

exponentiation <- a ^ b

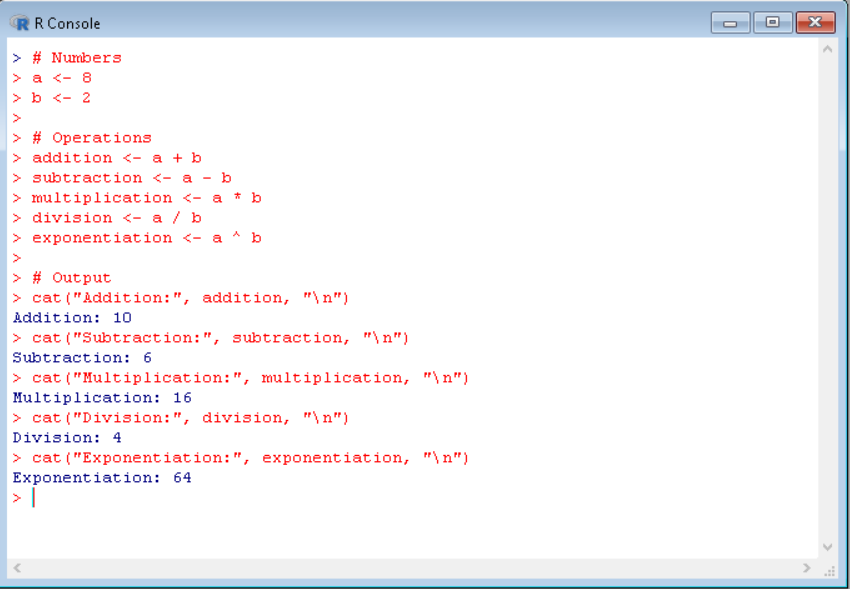
cat("Addition:", addition, "\n")

cat("Subtraction:", subtraction, "\n")

cat("Multiplication:", multiplication, "\n")

cat("Division:", division, "\n")

cat("Exponentiation:", exponentiation, "\n")



**2)** **DATA FRAME WITH EMPLOYEE DETAILS.**

**Aim:** Create a data frame with employee details (ID, Name, Age, Salary). Add a new column, filter rows, and sort by salary.

**Code:**

employee\_data <- data.frame(

ID = c(101, 102, 103, 104, 105),

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

Age = c(25, 30, 28, 35, 40),

Salary = c(50000, 60000, 55000, 70000, 80000)

)

employee\_data$Department <- c("HR", "Finance", "IT", "Admin", "Sales")

high\_salary <- subset(employee\_data, Salary > 60000)

sorted\_data <- employee\_data[order(employee\_data$Salary), ]

print("Original Data:")

print(employee\_data)

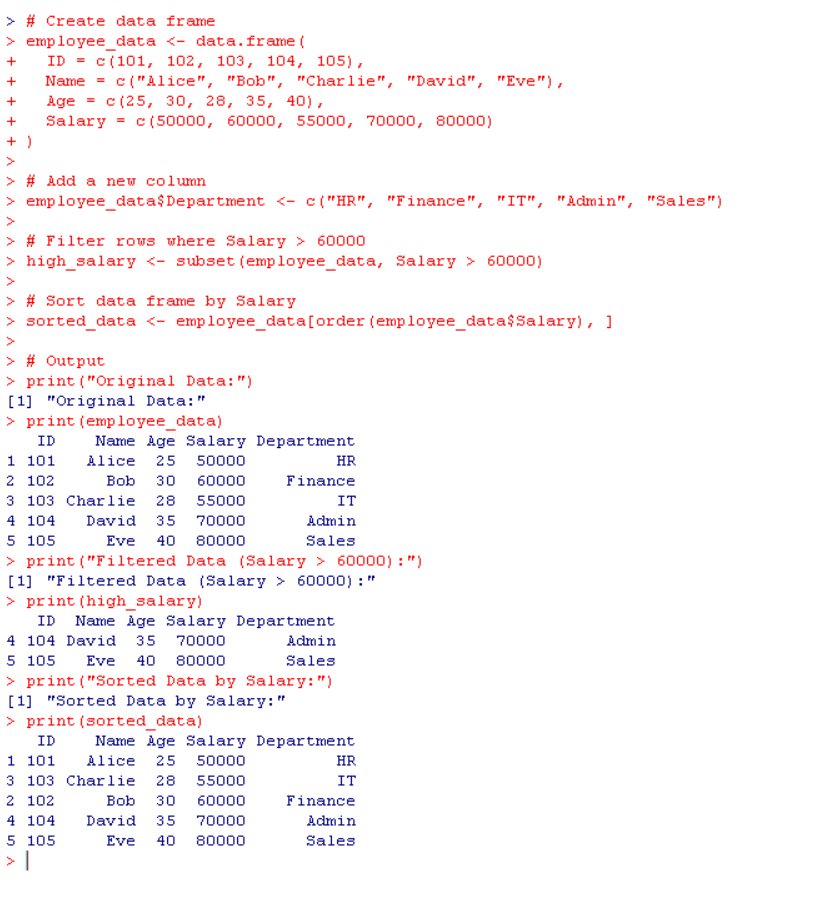
print("Filtered Data (Salary > 60000):")

print(high\_salary)

print("Sorted Data by Salary:")

print(sorted\_data)

**Output:**



**3) Perform slicing, reshaping, and calculating sums**

**Aim:** Perform slicing, reshaping, and calculate the sum of elements along dimensions.

**Code:**

# Create a matrix

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

# Slicing: Extract 2nd row and 3rd column

row\_2 <- matrix\_data[2, ]

col\_3 <- matrix\_data[, 3]

# Reshaping (transpose)

reshaped\_matrix <- t(matrix\_data)

# Sum along rows and columns

row\_sum <- rowSums(matrix\_data)

col\_sum <- colSums(matrix\_data)

**# Output:**

print("Original Matrix:")

print(matrix\_data)

print("2nd Row:")

print(row\_2)

print("3rd Column:")

print(col\_3)

print("Transposed Matrix:")

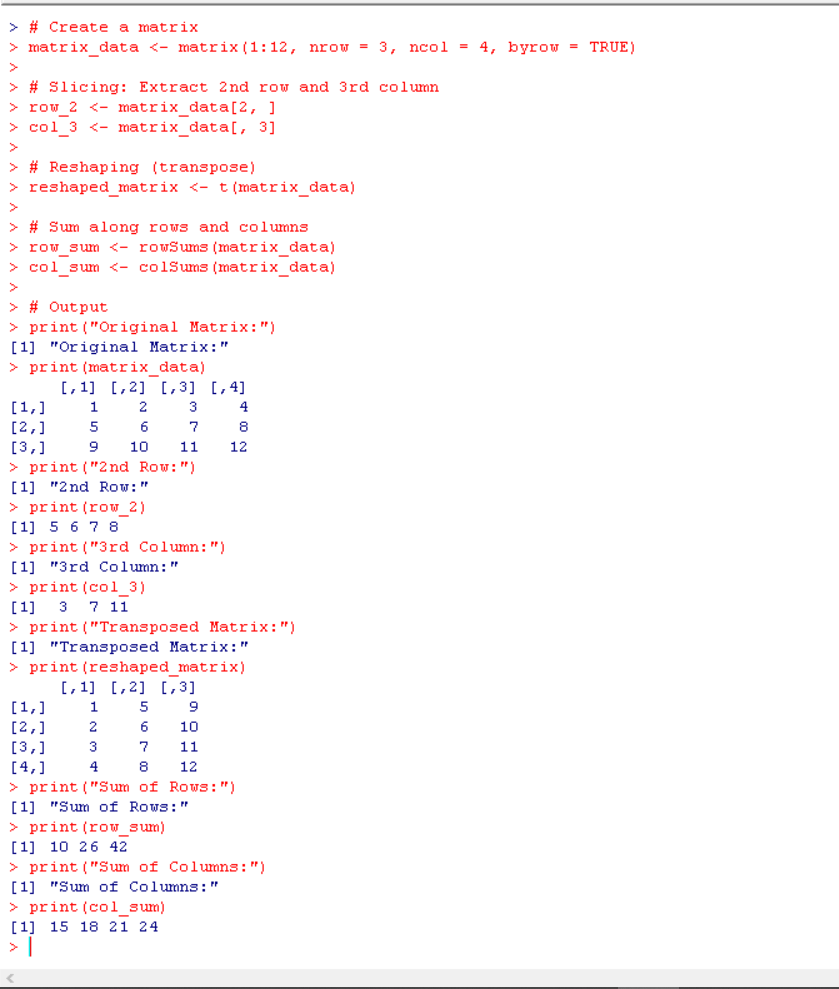
print(reshaped\_matrix)

print("Sum of Rows:")

print(row\_sum)

print("Sum of Columns:")

print(col\_sum)



**4) Sequence and mathematical operations**

**Aim:** Create a sequence from 20 to 50, find the mean of numbers 20–60, and the sum of numbers 51–91.

**Code:**

# Sequence

sequence <- 20:50

# Mean of numbers from 20 to 60

mean\_20\_to\_60 <- mean(20:60)

# Sum of numbers from 51 to 91

sum\_51\_to\_91 <- sum(51:91)

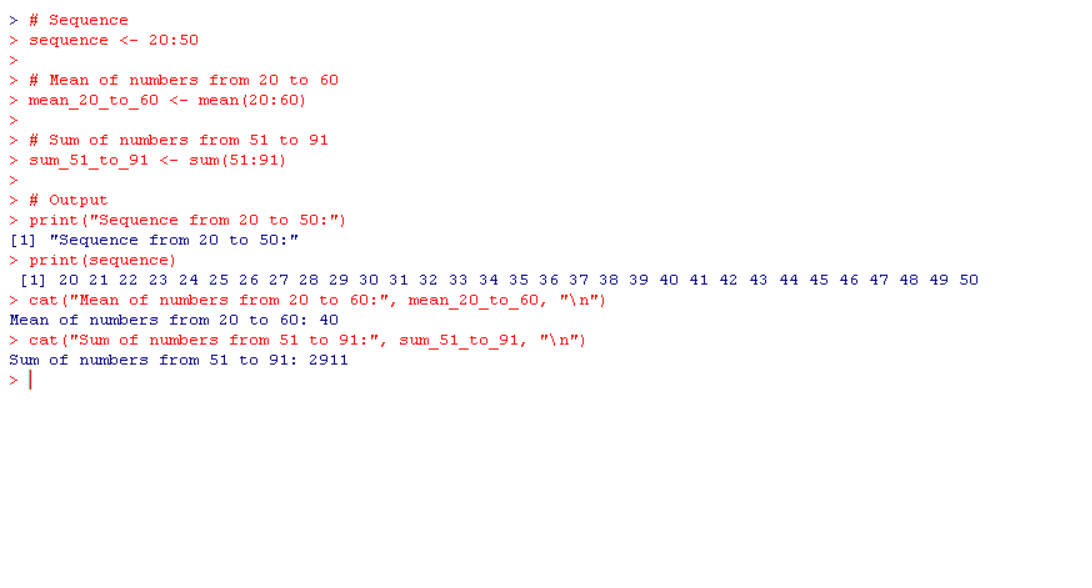
**# Output:**

print("Sequence from 20 to 50:")

print(sequence)

cat("Mean of numbers from 20 to 60:", mean\_20\_to\_60, "\n")

cat("Sum of numbers from 51 to 91:", sum\_51\_to\_91, "\n")



**5) Extract English letters**

**Aim:** Extract first 10 lowercase letters, last 10 uppercase letters, and letters between 22nd–24th in uppercase.

**Code:**

# Extract letters

first\_10 <- letters[1:10]

last\_10 <- LETTERS[17:26]

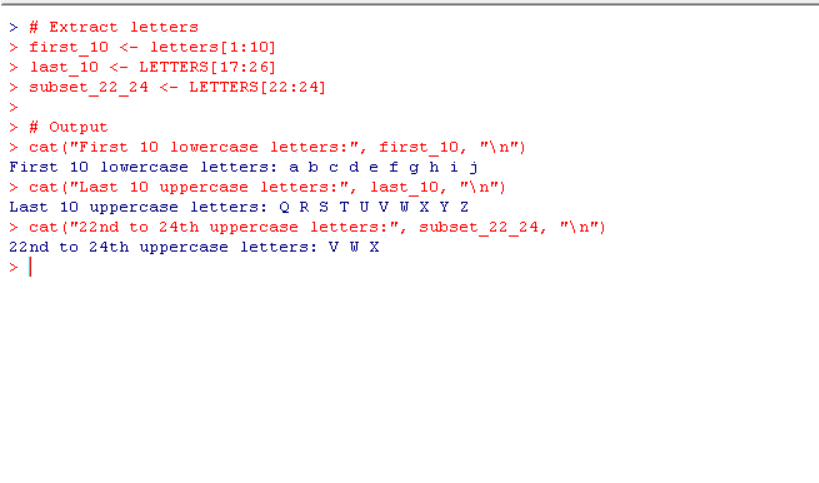
subset\_22\_24 <- LETTERS[22:24]

**# Output:**

cat("First 10 lowercase letters:", first\_10, "\n")

cat("Last 10 uppercase letters:", last\_10, "\n")

cat("22nd to 24th uppercase letters:", subset\_22\_24, "\n")



**6) Logical vector based on conditions**

**Aim:** Create a logical vector based on conditions applied to a numeric vector and use logical operations (AND, OR, NOT) for filtering.

**Code:**

# Numeric vector

numbers <- c(10, 15, 20, 25, 30, 35, 40)

greater\_than\_20 <- numbers > 20

less\_than\_35 <- numbers < 35

and\_condition <- greater\_than\_20 & less\_than\_35 # AND

or\_condition <- greater\_than\_20 | less\_than\_35 # OR

not\_condition <- !greater\_than\_20 # NOT

**# Output:**

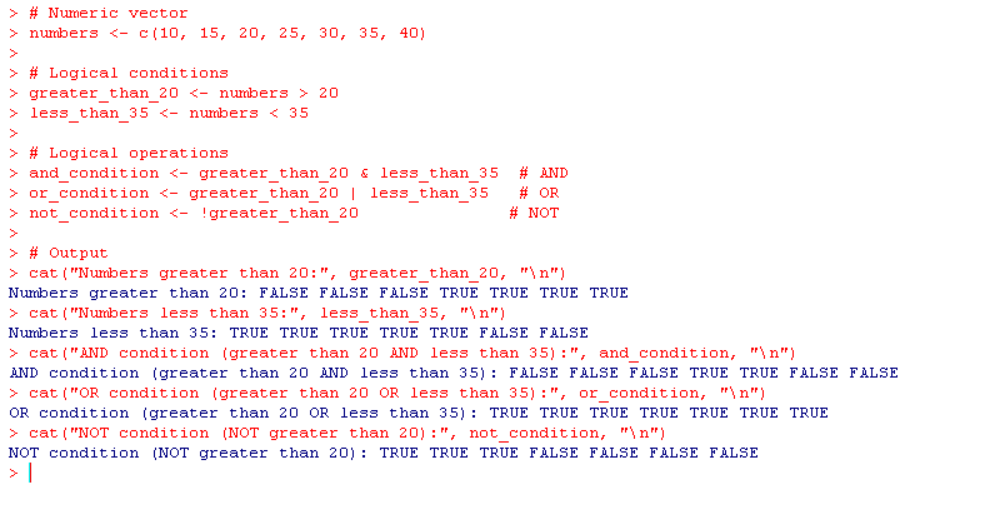
cat("Numbers greater than 20:", greater\_than\_20, "\n")

cat("Numbers less than 35:", less\_than\_35, "\n")

cat("AND condition (greater than 20 AND less than 35):", and\_condition, "\n")

cat("OR condition (greater than 20 OR less than 35):", or\_condition, "\n")

cat("NOT condition (NOT greater than 20):", not\_condition, "\n")



**7)Factor in R Script**

**Aim:** Create a factor from a character vector of categorical data (e.g., colors), explain its significance, and change the levels of a factor.

**Code:**

# Create a character vector

colors <- c("Red", "Blue", "Green", "Red", "Blue", "Green", "Red")

# Convert to factor

color\_factor <- factor(colors)

# Significance: Factors are used for categorical data in statistical analysis,

# enabling efficient storage and computation.

# Change levels

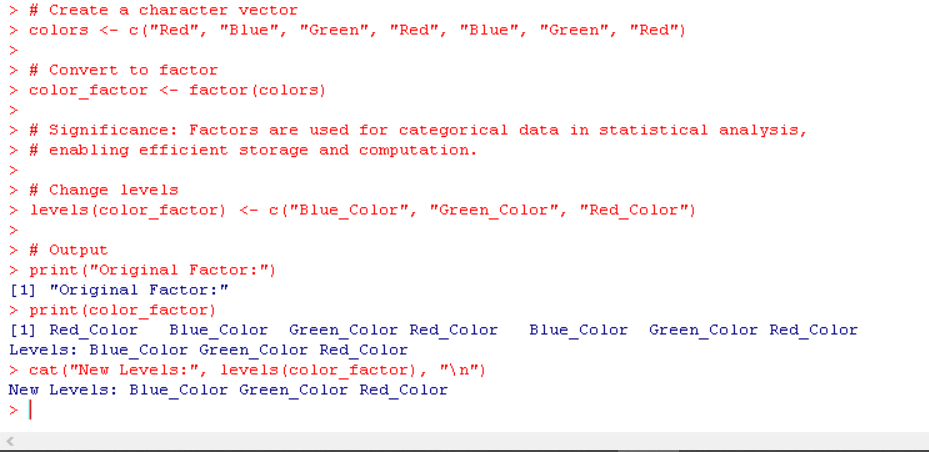
levels(color\_factor) <- c("Blue\_Color", "Green\_Color", "Red\_Color")

# Output

print("Original Factor:")

print(color\_factor)

cat("New Levels:", levels(color\_factor), "\n")



**8)Data types in R and type conversion**

**Aim:** Discuss data types in R and demonstrate type conversion between them.

**Code:**

# Examples of data types

numeric\_value <- 10.5 # Numeric

integer\_value <- as.integer(10) # Integer

character\_value <- "Hello" # Character

logical\_value <- TRUE # Logical

complex\_value <- 2 + 3i # Complex

# Type conversion

num\_to\_char <- as.character(numeric\_value) # Numeric to Character

char\_to\_num <- as.numeric("100") # Character to Numeric

log\_to\_int <- as.integer(logical\_value) # Logical to Integer

**# Output:**

cat("Numeric:", numeric\_value, "\n")

cat("Integer:", integer\_value, "\n")

cat("Character:", character\_value, "\n")

cat("Logical:", logical\_value, "\n")

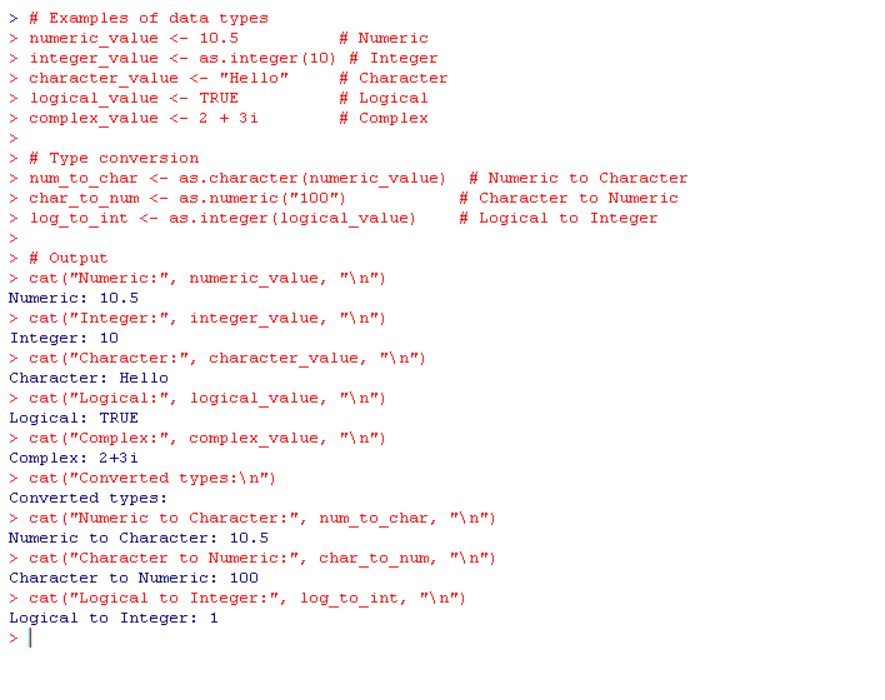
cat("Complex:", complex\_value, "\n")

cat("Converted types:\n")

cat("Numeric to Character:", num\_to\_char, "\n")

cat("Character to Numeric:", char\_to\_num, "\n")

cat("Logical to Integer:", log\_to\_int, "\n")

**9)Create and fill matrices**

**Aim:** Create a 5 × 4 matrix, a 3 × 3 matrix (filled by rows with labels), and a 2 × 2 matrix (filled by columns with labels).

**Code:**

# 5 × 4 matrix

matrix\_5x4 <- matrix(1:20, nrow = 5, ncol = 4)

# 3 × 3 matrix filled by rows

matrix\_3x3 <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)

rownames(matrix\_3x3) <- c("Row1", "Row2", "Row3")

colnames(matrix\_3x3) <- c("Col1", "Col2", "Col3")

# 2 × 2 matrix filled by columns

matrix\_2x2 <- matrix(1:4, nrow = 2, ncol = 2, byrow = FALSE)

rownames(matrix\_2x2) <- c("RowA", "RowB")

colnames(matrix\_2x2) <- c("ColX", "ColY")

**# Output:**

cat("5 × 4 Matrix:\n")

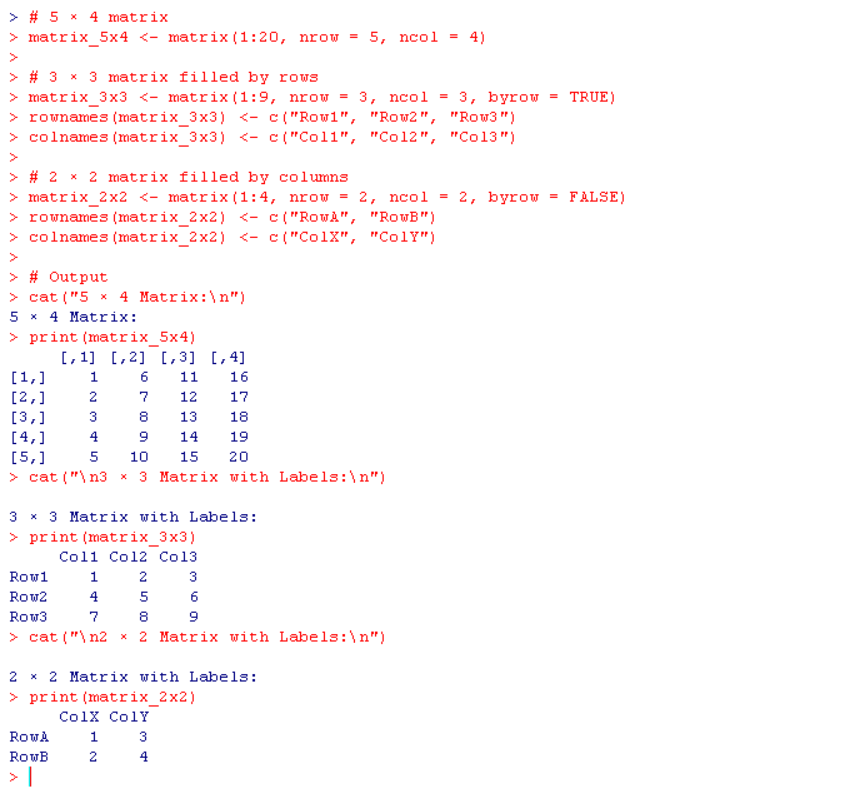
print(matrix\_5x4)

cat("\n3 × 3 Matrix with Labels:\n")

print(matrix\_3x3)

cat("\n2 × 2 Matrix with Labels:\n")

print(matrix\_2x2)



**10) Create a 5 × 3 array**

**Aim:** Create a two-dimensional 5 × 3 array of sequences of even integers greater than 50.

**Code:**

# Create two vectors of even integers greater than 50

vector1 <- seq(52, 72, by = 2)

vector2 <- seq(74, 94, by = 2)

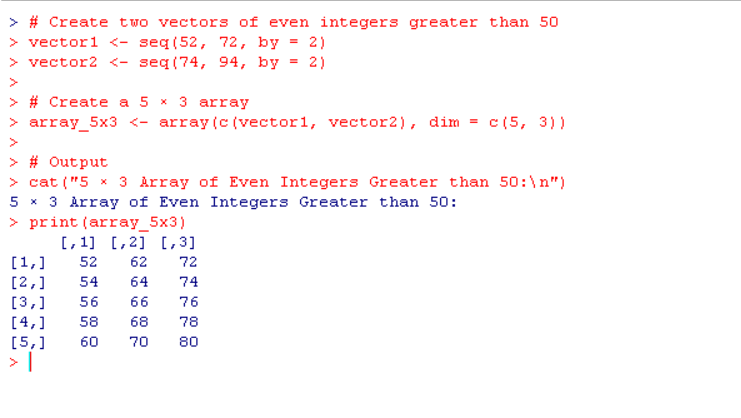
# Create a 5 × 3 array

array\_5x3 <- array(c(vector1, vector2), dim = c(5, 3))

**# Output:**

cat("5 × 3 Array of Even Integers Greater than 50:\n")

print(array\_5x3)



**11)** **Access Values in a Vector and Print the Vector**

**AIM**: To create a vector, access its elements, and print the vector.

**# CODE:**

# Creating a vector

my\_vector <- c(10, 20, 30, 40, 50)

# Accessing values in the vector

first\_element <- my\_vector[1] # First element

last\_element <- my\_vector[length(my\_vector)] # Last element

# Printing the vector and accessed values

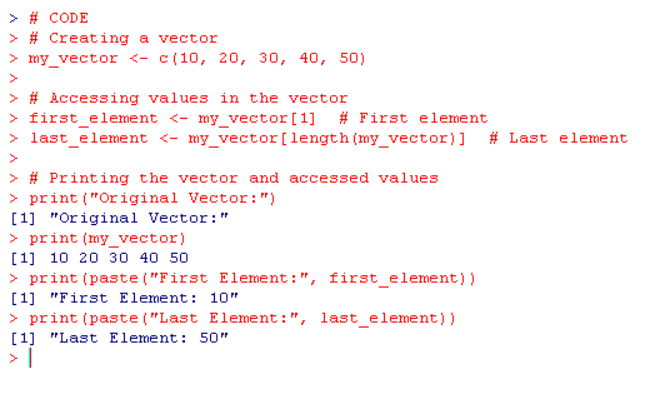
print("Original Vector:")

print(my\_vector)

print(paste("First Element:", first\_element))

print(paste("Last Element:", last\_element))

**OUTPUT**:



**12) Find the Nth Smallest Value in a Vector**

**AIM**: To find the Nth smallest value in a vector.

**CODE:**

# Creating a vector

my\_vector <- c(5, 2, 9, 1, 7)

# Specify the value of N

N <- 3

# Finding the Nth smallest value

nth\_smallest <- sort(my\_vector)[N]

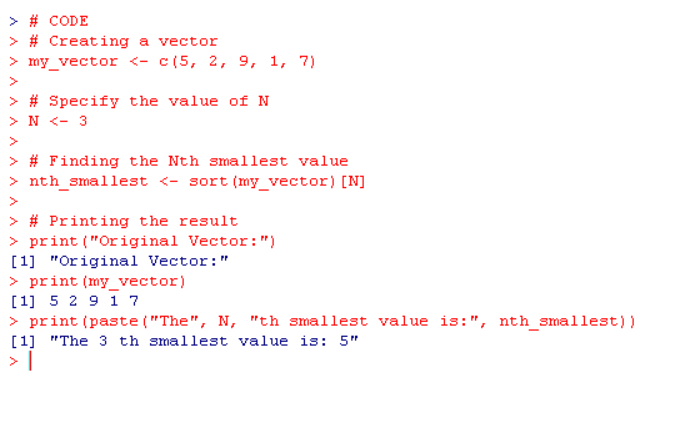
# Printing the result

print("Original Vector:")

print(my\_vector)

print(paste("The", N, "th smallest value is:", nth\_smallest))

**OUTPUT:**

****

**13)Concatenate a Vector of Strings**

**AIM: To concatenate a vector of strings into a single string.**

**CODE:**

# Creating a vector of strings

string\_vector <- c("Hello", "World", "in", "R")

# Concatenating strings with a separator

concatenated\_string <- paste(string\_vector, collapse = " ")

# Printing the concatenated string

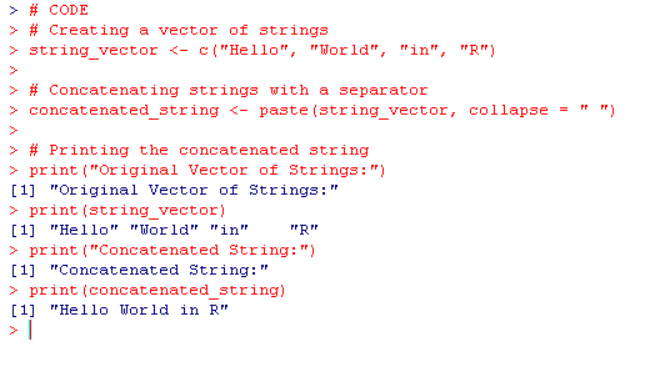
print("Original Vector of Strings:")

print(string\_vector)

print("Concatenated String:")

print(concatenated\_string)

**OUTPUT:**

****

**14)Find the Row and Column Index of Maximum and Minimum Value in a Matrix**

**AIM: To find the row and column index of the maximum and minimum values in a matrix.**

**CODE:**

# Creating a matrix

my\_matrix <- matrix(c(3, 7, 1, 8, 4, 9), nrow = 2, byrow = TRUE)

# Finding max and min values and their indices

max\_value <- max(my\_matrix)

max\_index <- which(my\_matrix == max\_value, arr.ind = TRUE)

min\_value <- min(my\_matrix)

min\_index <- which(my\_matrix == min\_value, arr.ind = TRUE)

# Printing the matrix and indices

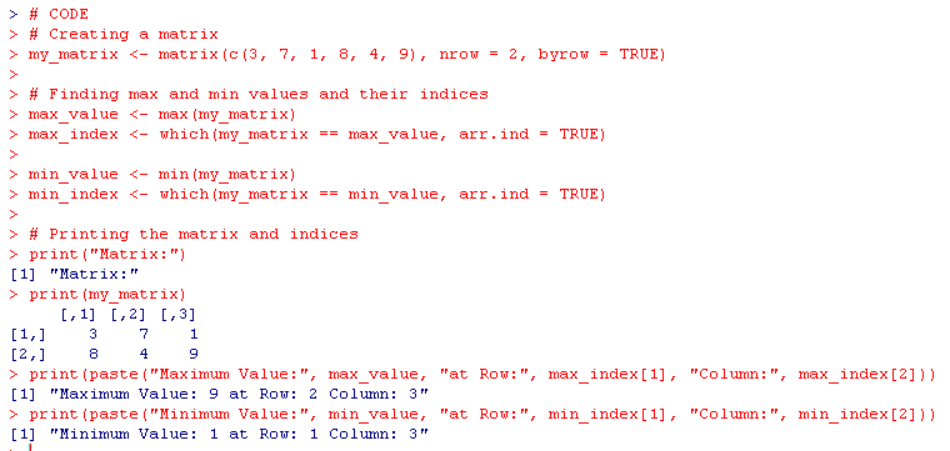
print("Matrix:)

print(my\_matrix)

print(paste("Maximum Value:", max\_value, "at Row:", max\_index[1], "Column:", max\_index[2]))

print(paste("Minimum Value:", min\_value, "at Row:", min\_index[1], "Column:", min\_index[2]))

**OUTPUT:**

****

**15)Print Numbers 1 to 100 with FizzBuzz**

**AIM: To print numbers from 1 to 100 with specific substitutions for multiples of 3, 5, or both.**

**CODE:**

# Loop through numbers 1 to 100

for (i in 1:100) {

if (i %% 3 == 0 && i %% 5 == 0) {

print("FizzBuzz")

} else if (i %% 3 == 0) {

print("Fizz")

} else if (i %% 5 == 0) {

print("Buzz")

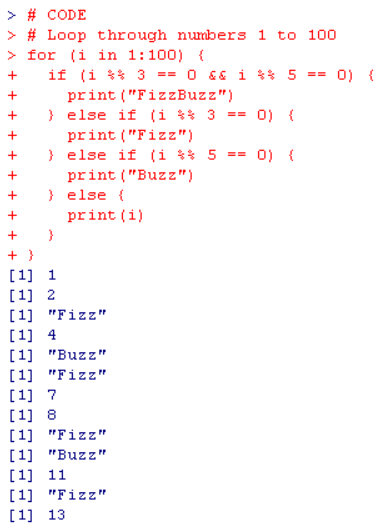
} else {

print(i)

}

}

**OUTPUT:**

****

**16)Convert a List to a Data Frame with Specific Column Names**

**AIM: To convert a list into a data frame with specific column names.**

**CODE:**

# Creating a list

my\_list <- list(

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

Age = c(25, 30, 35),

Salary = c(50000, 60000, 70000)

)

# Converting the list to a data frame

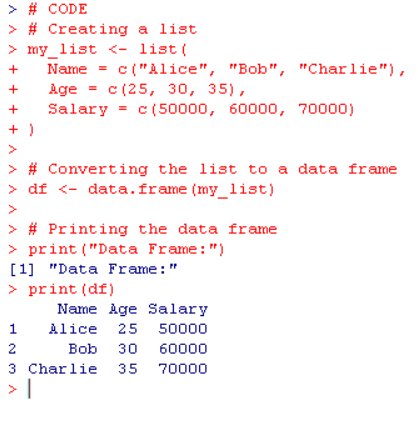
df <- data.frame(my\_list)

# Printing the data frame

print("Data Frame:")

print(df)

**OUTPUT:**

****

**17) Create a Data Frame for 5 Employees and Display a Summary**

**AIM: To create a data frame containing details of 5 employees and display a summary of the data.**

**CODE:**

# Creating a data frame

employee\_data <- data.frame(

EmployeeID = 1:5,

Name = c("John", "Jane", "Smith", "Emily", "Michael"),

Age = c(28, 34, 45, 30, 40),

Department = c("HR", "IT", "Finance", "Marketing", "Operations"),

Salary = c(50000, 70000, 65000, 60000, 75000)

)

# Printing the data frame

print("Employee Data:")

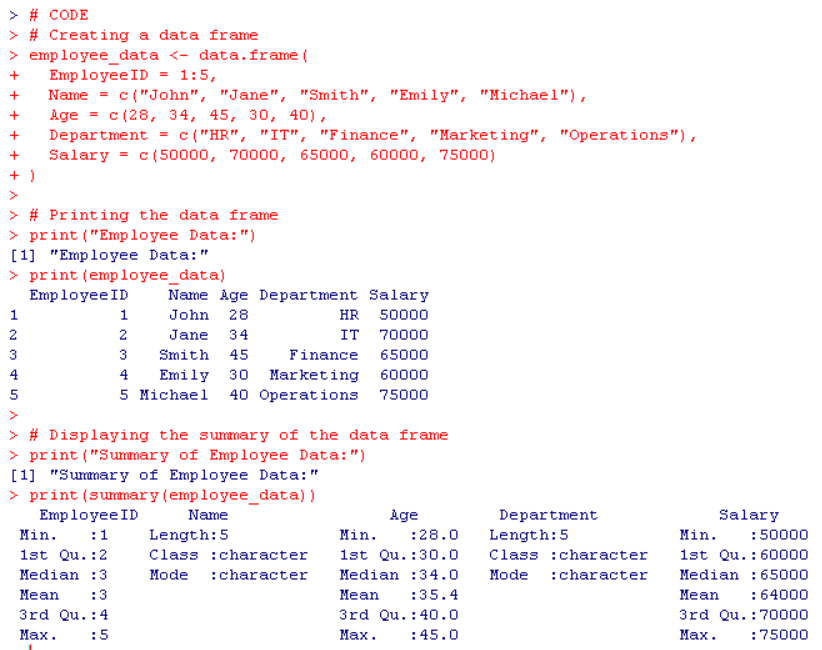
print(employee\_data)

# Displaying the summary of the data frame

print("Summary of Employee Data:")

print(summary(employee\_data))

**OUTPUT:**

****

**18) Find the Maximum and Minimum Value of a Vector**

**AIM: To find the maximum and minimum values in a given vector.**

**CODE:**

# Creating a vector

my\_vector <- c(12, 45, 7, 89, 23, 67)

# Finding the maximum and minimum values

max\_value <- max(my\_vector)

min\_value <- min(my\_vector)

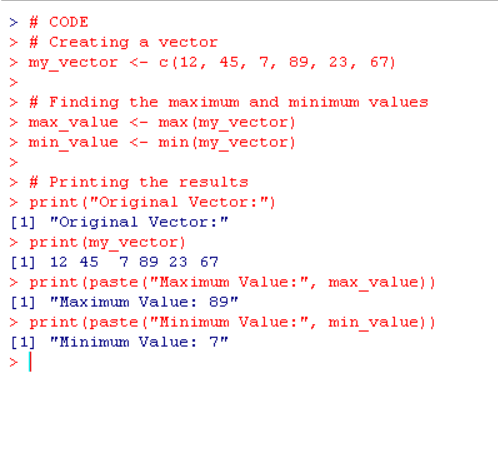
# Printing the results

print("Original Vector:")

print(my\_vector)

print(paste("Maximum Value:", max\_value))

print(paste("Minimum Value:", min\_value)

**OUTPUT:  
**

**19) Create a 3x3x2 Array with Two Vectors**

**AIM: To create a 3x3x2 array from two vectors and print the array.**

**CODE:**

# Creating two vectors

vector1 <- c(1, 2, 3, 4, 5, 6)

vector2 <- c(7, 8, 9, 10, 11, 12)

# Creating the array

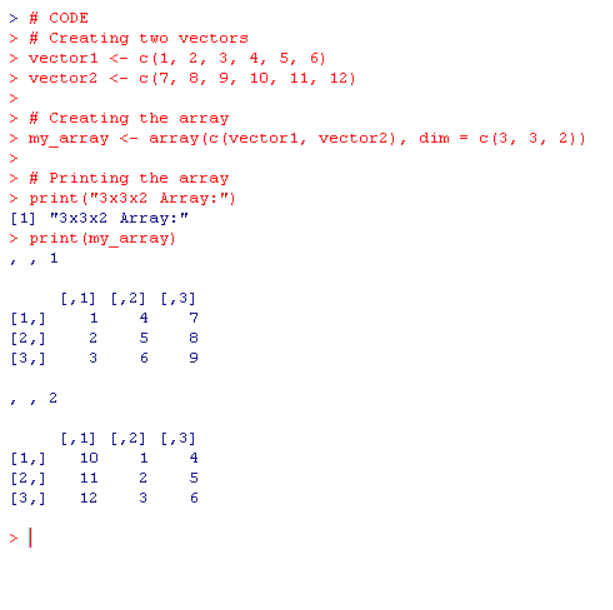
my\_array <- array(c(vector1, vector2), dim = c(3, 3, 2)

# Printing the array

print("3x3x2 Array:")

print(my\_array)

**OUTPUT:**

****

**20) Assign Grades Based on Student Scores Using If-Else**

**AIM: To assign grades to students based on their scores using an if-else statement.**

**CODE:**

# Creating a vector of scores

scores <- c(85, 72, 58, 90, 49)

# Function to assign grades

assign\_grade <- function(score) {

if (score >= 85) {

return("A")

} else if (score >= 70) {

return("B")

} else if (score >= 50) {

return("C")

} else {

return("F")

}

}

# Applying the function to assign grades

grades <- sapply(scores, assign\_grade)

# Printing scores and their corresponding grades

print("Scores and Grades:")

print(data.frame(Scores = scores, Grades = grades))

**OUTPUT:**

