NAME – SAARA ANAND

REG NO – 21BCE8156

SLOT – L55+L56

FDA LAB ASSIGNMENT 3-

Vectors

1. **Create vector of numeric, complex, logical and character types of length 5.**

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

numeric_vector

class(numeric_vector)

complex_vector <- c(1+3i, 3+4i, 5+3i, 7+8i, 9+10i)

complex_vector

class(complex_vector)

logical_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)

logical_vector

class(logical_vector)

character_vector <- c("apple", "banana", "cherry", "grape",
"kiwi")

character_vector

class(character_vector)
```

```
> numeric_vector <- c(1, 2, 3, 4, 5)
> numeric_vector
[1] 1 2 3 4 5
> class(numeric_vector)
[1] "numeric"
> complex_vector <- c(1+3i, 3+4i, 5+3i, 7+8i, 9+10i)
> complex_vector
[1] 1+ 3i 3+ 4i 5+ 3i 7+ 8i 9+10i
> class(complex_vector)
[1] "complex"
> logical_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
> logical_vector
[1]  TRUE FALSE  TRUE  TRUE FALSE
> class(logical_vector)
[1] "logical"
> character_vector <- c("apple", "banana", "cherry", "grape", "kiwi")
> character_vector
[1] "apple"  "banana" "cherry" "grape"  "kiwi"
> class(character_vector)
[1] "character"
>
```

2. **Write a R program to add, multiply & divide two vectors of integers type and length 4.**

```
# Define the two vectors

vector1 <- c(1, 3, 5, 7)

vector2 <- c(5, 7, 9, 11)

# Addition

addition_result <- vector1 + vector2

cat("Addition Result: ", addition_result, "\n")

# Multiplication

multiplication_result <- vector1 * vector2

cat("Multiplication Result: ", multiplication_result, "\n")

# Division
```

division_result <- vector1 / vector2

cat("Division Result: ", division_result, "\n")

```
> # Define the two vectors
> vector1 <- c(1, 3, 5, 7)
> vector2 <- c(5, 7, 9, 11)
> # Addition
> addition_result <- vector1 + vector2
> cat("Addition Result: ", addition_result, "\n")
Addition Result:  6 10 14 18
> # Multiplication
> multiplication_result <- vector1 * vector2
> cat("Multiplication Result: ", multiplication_result, "\n")
Multiplication Result:  5 21 45 77
> # Division
> division_result <- vector1 / vector2
> cat("Division Result: ", division_result, "\n")
Division Result:  0.2 0.4285714 0.5555556 0.6363636
>
```

3. **Write a R program to append value to a given empty vector.**

Create an empty vector

my_vector <- vector()

Append a value to the vector

my_vector <- c(my_vector, 5)

Print the vector

print(my_vector)

```
> # Create an empty vector
> my_vector <- vector()
> # Append a value to the vector
> my_vector <- c(my_vector, 5)
> # Print the vector
> print(my_vector)
[1] 5
>
```

4. **Write a R program to find Sum, Mean and Product of a Vector.**

```
# Define the vector

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

# Calculate the sum

sum_result <- sum(my_vector)

cat("Sum: ", sum_result, "\n")

# Calculate the mean

mean_result <- mean(my_vector)

cat("Mean: ", mean_result, "\n")

# Calculate the product

product_result <- prod(my_vector)

cat("Product: ", product_result, "\n")
```

```
> # Define the vector
> my_vector <- c(1, 2, 3, 4, 5)
> # Calculate the sum
> sum_result <- sum(my_vector)
> cat("Sum: ", sum_result, "\n")
Sum:  15
> # Calculate the mean
> mean_result <- mean(my_vector)
> cat("Mean: ", mean_result, "\n")
Mean:  3
> # Calculate the product
> product_result <- prod(my_vector)
> cat("Product: ", product_result, "\n")
Product:  120
>
```

5. **Write a R program to find Sum, Mean and Product of a Vector, ignore element like NA or NaN.**

```
# Define a vector with NA and NaN elements

my_vector <- c(1, 2, NA, 4, NaN, 6)

# Sum, ignoring NA and NaN

sum_result <- sum(my_vector, na.rm = TRUE)

print(sum_result)

# Mean, ignoring NA and NaN

mean_result <- mean(my_vector, na.rm = TRUE)

print(mean_result)

# Product, ignoring NA and NaN

product_result <- prod(my_vector, na.rm = TRUE)

print(product_result)
```

```
> # Define a vector with NA and NaN elements
> my_vector <- c(1, 2, NA, 4, NaN, 6)
> # Sum, ignoring NA and NaN
> sum_result <- sum(my_vector, na.rm = TRUE)
> print(sum_result)
[1] 13
> # Mean, ignoring NA and NaN
> mean_result <- mean(my_vector, na.rm = TRUE)
> print(mean_result)
[1] 3.25
> # Product, ignoring NA and NaN
> product_result <- prod(my_vector, na.rm = TRUE)
> print(product_result)
[1] 48
>
```

6. Write a R program to find the minimum and the maximum of a Vector.

```
# Define the vector

vector <- c(56, 21, 83, 16, 98)

# Find the minimum value

min_value <- min(vector)

cat("Minimum value: ", min_value, "\n")

# Find the maximum value

max_value <- max(vector)

cat("Maximum value: ", max_value, "\n")
```

```
> # Define the vector
> vector <- c(56, 21, 83, 16, 98)
> # Find the minimum value
> min_value <- min(vector)
> cat("Minimum value: ", min_value, "\n")
Minimum value:  16
> # Find the maximum value
> max_value <- max(vector)
> cat("Maximum value: ", max_value, "\n")
Maximum value:  98
>
```

7. Write a R program to sort a Vector in ascending and descending order.

```
# Define the vector

vector <- c(54, 25, 89, 15, 92)

# Sort in ascending order

ascending_order <- sort(vector)

cat("Ascending order: ", ascending_order, "\n")

# Sort in descending order

descending_order <- sort(vector, decreasing = TRUE)

cat("Descending order: ", descending_order, "\n")
```

```
> # Define the vector
> vector <- c(54, 25, 89, 15, 92)
> # Sort in ascending order
> ascending_order <- sort(vector)
> cat("Ascending order: ", ascending_order, "\n")
Ascending order:  15 25 54 89 92
> # Sort in descending order
> descending_order <- sort(vector, decreasing = TRUE)
> cat("Descending order: ", descending_order, "\n")
Descending order:  92 89 54 25 15
>
```

8. Write a R program to test whether a given vector contains a specified element.

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

element_to_find <- 4

contains_element <- element_to_find %in% vector

if (contains_element) {

  cat("The vector contains the element ", element_to_find, "\n")

} else {

  cat("The vector does not contain the element ", element_to_find, "\n")

}
```

```
> vector <- c(1, 2, 3, 4, 5)
> element_to_find <- 4
> contains_element <- element_to_find %in% vector
> if (contains_element) {
+   cat("The vector contains the element ", element_to_find, "\n")
+ } else {
+   cat("The vector does not contain the element ", element_to_find, "\n")
+ }
The vector contains the element  4
>
```

9. Write a R program to find nth highest value in a given vector.

```
# Define the vector

my_vector <- c(5, 2, 8, 1, 9)

# Specify the value of n
```

```
n <- 3

# Find the nth highest value

sorted_vector <- sort(my_vector, decreasing = TRUE)

nth_highest_value <- sorted_vector[n]

# Print the result

cat("The", n, "highest value in the vector is:", nth_highest_value,
"\n")
```

```
> # Define the vector
> my_vector <- c(5, 2, 8, 1, 9)
> # Specify the value of n
> n <- 3
> # Find the nth highest value
> sorted_vector <- sort(my_vector, decreasing = TRUE)
> nth_highest_value <- sorted_vector[n]
> # Print the result
> cat("The", n, "highest value in the vector is:", nth_highest_value, "\n")
The 3 highest value in the vector is: 5
>
```

10. Write a R program to create a vector using : operator and seq() function.

```
# Using the : operator

vector1 <- 1:5

cat("Vector using : operator: ", vector1, "\n")

# Using the seq() function

vector2 <- seq(from = 1, to = 10, by = 2)
```

cat("Vector using seq() function: ", vector2, "\n")

```
> # Using the : operator
> vector1 <- 1:5
> cat("Vector using : operator: ", vector1, "\n")
Vector using : operator:  1 2 3 4 5
> # Using the seq() function
> vector2 <- seq(from = 1, to = 10, by = 2)
> cat("Vector using seq() function: ", vector2, "\n")
Vector using seq() function:  1 3 5 7 9
>
```

Lists

1. **Write a R program to create a list containing strings, numbers, vectors and a logical values.**

```
my_list <- list(
  my_string = "Hello, world, hi!",
  my_number = 89,
  my_vector = c(1, 2, 3, 4, 5),
  my_logical = TRUE
)
print(my_list)
```

```
> my_list <- list(
+   my_string = "Hello, world, hi!",
+   my_number = 89,
+   my_vector = c(1, 2, 3, 4, 5),
+   my_logical = TRUE
+ )
> print(my_list)
$my_string
[1] "Hello, world, hi!"

$my_number
[1] 89

$my_vector
[1] 1 2 3 4 5

$my_logical
[1] TRUE

>
```

2. **If Newlist <- list(a=1:10, b="Good morning", c="Hi"), write an R statement that will add 1 to each element of the first vector in Newlist.**

   Newlist <- list(a=1:10, b="Good morning", c="Hi")

   Newlist$a <- Newlist$a + 1

   Newlist

```
> Newlist <- list(a=1:10, b="Good morning", c="Hi")
> Newlist$a <- Newlist$a + 1
> Newlist
$a
 [1]  2  3  4  5  6  7  8  9 10 11

$b
[1] "Good morning"

$c
[1] "Hi"

>
```

3. **Consider y <- list("a", "b", "c"), write an R statement that will assign new names "one", "two" and "three" to the elements of y.**

```
y <- list("a", "b", "c")

names(y) <- c("one", "two", "three")

print(y$one)

print(y$two)

print(y$three)
```

```
> y <- list("a", "b", "c")
> names(y) <- c("one", "two", "three")
> print(y$one)
[1] "a"
> print(y$two)
[1] "b"
> print(y$three)
[1] "c"
>
```

4. **Let string <- "Grand Opening", write an R statement to split this string into two and return the following output: "Grand" "Opening".**

```
string <- "Grand Opening"

split_string <- strsplit(string, " ")[[1]]

print(split_string[1])

print(split_string[2])
```

```
> string <- "Grand Opening"
> split_string <- strsplit(string, " ")[[1]]
> print(split_string[1])
[1] "Grand"
> print(split_string[2])
[1] "Opening"
>
```

5. **Write a R program to select second element of a given nested list.**

```
# Define the nested list

nested_list <- list(

  a = list(1, 2, 3),

  b = list("red", "blue", "pink"),

  c = list(TRUE, FALSE, TRUE)

)

# Select the second element of the nested list

second_element <- nested_list[[2]]

# Print the second element

print(second_element)
```

```
> # Define the nested list
> nested_list <- list(
+   a = list(1, 2, 3),
+   b = list("red", "blue", "pink"),
+   c = list(TRUE, FALSE, TRUE)
+ )
> # Select the second element of the nested list
> second_element <- nested_list[[2]]
> # Print the second element
> print(second_element)
[[1]]
[1] "red"

[[2]]
[1] "blue"

[[3]]
[1] "pink"

>
```

6. **Write a R program to merge two given lists into one list.**

list1 <- list(a = 1, b = 2, c = 3)

list2 <- list(d = 4, e = 5, f = 6)

merged_list <- c(list1, list2)

print(merged_list)

```
> list1 <- list(a = 1, b = 2, c = 3)
> list2 <- list(d = 4, e = 5, f = 6)
> merged_list <- c(list1, list2)
> print(merged_list)
$a
[1] 1

$b
[1] 2

$c
[1] 3

$d
[1] 4

$e
[1] 5

$f
[1] 6

>
```

7. **Write a R program to convert a given list to vector.**

```
my_list <- list("red", "pink", "white")

my_vector <- unlist(my_list)

print(my_vector)
```

```
> my_list <- list("red", "pink", "white")
> my_vector <- unlist(my_list)
> print(my_vector)
[1] "red"   "pink"  "white"
>
```

8. **Write a R program to add a new item a = "R Programming" to a given list.**

```
# Define the original list

my_list <- list(b = 1, c = 2, d = 3)

# Add a new item to the list

my_list$a <- "R Programming"

# Print the updated list

print(my_list)
```

```
> # Define the original list
> my_list <- list(b = 1, c = 2, d = 3)
> # Add a new item to the list
> my_list$a <- "R Programming"
> # Print the updated list
> print(my_list)
$b
[1] 1

$c
[1] 2

$d
[1] 3

$a
[1] "R Programming"

>
```

9. **Write a R program to get the length of the first two vectors of a given list.**

```
# Define the list

my_list <- list(a = c(1, 2, 3, 4, 5), b = c("apple", "banana", "cherry"), c = c(TRUE, FALSE, TRUE))

# Get the length of the first two vectors

length_vector1 <- length(my_list[[1]])

length_vector2 <- length(my_list[[2]])

# Print the lengths

print(length_vector1)

print(length_vector2)
```

```
> # Define the list
> my_list <- list(a = c(1, 2, 3, 4, 5), b = c("apple", "banana", "cherry"), c = c(TRUE, FALSE, TRU
E))
> # Get the length of the first two vectors
> length_vector1 <- length(my_list[[1]])
> length_vector2 <- length(my_list[[2]])
> # Print the lengths
> print(length_vector1)
[1] 5
> print(length_vector2)
[1] 3
>
```

10. **Write a R program to find all elements of a given list that are not in another given list.**

```
# Define the first list
```

```
list1 <- list("green", "blue", "red", "orange")

# Define the second list

list2 <- list("red", "white", "pink")

# Find elements in list1 not present in list2

not_in_list2 <- list1[!list1 %in% list2]

# Print the elements not in list2

print(not_in_list2)
```

```
> # Define the first list
> list1 <- list("green", "blue", "red", "orange")
> # Define the second list
> list2 <- list("red", "white", "pink")
> # Find elements in list1 not present in list2
> not_in_list2 <- list1[!list1 %in% list2]
> # Print the elements not in list2
> print(not_in_list2)
[[1]]
[1] "green"

[[2]]
[1] "blue"

[[3]]
[1] "orange"

>
```

Matrices

1. **Write a R program to create a matrix taking a given vector of numbers as input and define the column and row names. Display the matrix.**

```
# Define the vector of numbers
```

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

# Define the column names

column_names <- c("A", "B")

# Define the row names

row_names <- c("Row1", "Row2", "Row3")

# Create the matrix

my_matrix <- matrix(input_vector, nrow = length(row_names),
ncol = length(column_names), byrow = TRUE)

# Set the column and row names

colnames(my_matrix) <- column_names

rownames(my_matrix) <- row_names

# Display the matrix

print(my_matrix)
```

```
> # Define the vector of numbers
> input_vector <- c(1, 2, 3, 4, 5, 6)
> # Define the column names
> column_names <- c("A", "B")
> # Define the row names
> row_names <- c("Row1", "Row2", "Row3")
> # Create the matrix
> my_matrix <- matrix(input_vector, nrow = length(row_names), ncol = length(column_names), byrow = TRUE)
> # Set the column and row names
> colnames(my_matrix) <- column_names
> rownames(my_matrix) <- row_names
> # Display the matrix
> print(my_matrix)
     A B
Row1 1 2
Row2 3 4
Row3 5 6
>
```

2. **Write a R program to access the element at 3rd column and 2nd row, only the 3rd row and only the 4th column of a given matrix.**

```
# Define the matrix

my_matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)

# Access the element at 3rd column and 2nd row

element_3_2 <- my_matrix[2, 3]

# Access only the 3rd row

row_3 <- my_matrix[3, ]

# Access only the 4th column

column_4 <- my_matrix[, 4]

# Print the accessed elements

print(element_3_2)

print(row_3)

print(column_4)
```

```
> # Define the matrix
> my_matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)
> # Access the element at 3rd column and 2nd row
> element_3_2 <- my_matrix[2, 3]
> # Access only the 3rd row
> row_3 <- my_matrix[3, ]
> # Access only the 4th column
> column_4 <- my_matrix[ ,4]
Error in my_matrix[, 4] : subscript out of bounds
>
```

3.Write a R program to create two 2x3 matrix and add, subtract, multiply and divide the matrixes.

```
# Create the first matrix

matrix1 <- matrix(c(10, 20, 30, 40, 50, 60), nrow = 2, ncol = 3, byrow = TRUE)

print("Matrix 1:")

print(matrix1)

# Create the second matrix

matrix2 <- matrix(c(70, 80, 90, 100, 110, 120), nrow = 2, ncol = 3, byrow = TRUE)

print("Matrix 2:")

print(matrix2)

# Addition of matrices

matrix_addition <- matrix1 + matrix2

print("Addition:")

print(matrix_addition)

# Subtraction of matrices

matrix_subtraction <- matrix1 - matrix2

print("Subtraction:")

print(matrix_subtraction)

# Multiplication of matrices

matrix_multiplication <- matrix1 * matrix2
```

print("Multiplication:")

print(matrix_multiplication)

Division of matrices

matrix_division <- matrix1 / matrix2

print("Division:")

print(matrix_division)

```
> # Create the first matrix
> matrix1 <- matrix(c(10, 20, 30, 40, 50, 60), nrow = 2, ncol = 3, byrow = TRUE)
> print("Matrix 1:")
[1] "Matrix 1:"
> print(matrix1)
     [,1] [,2] [,3]
[1,]   10   20   30
[2,]   40   50   60
> # Create the second matrix
> matrix2 <- matrix(c(70, 80, 90, 100, 110, 120), nrow = 2, ncol = 3, byrow = TRUE)
> print("Matrix 2:")
[1] "Matrix 2:"
> print(matrix2)
     [,1] [,2] [,3]
[1,]   70   80   90
[2,]  100  110  120
> # Addition of matrices
> matrix_addition <- matrix1 + matrix2
> print("Addition:")
[1] "Addition:"
> print(matrix_addition)
     [,1] [,2] [,3]
[1,]   80  100  120
[2,]  140  160  180

> # Subtraction of matrices
> matrix_subtraction <- matrix1 - matrix2
> print("Subtraction:")
[1] "Subtraction:"
> print(matrix_subtraction)
     [,1] [,2] [,3]
[1,]  -60  -60  -60
[2,]  -60  -60  -60
> # Multiplication of matrices
> matrix_multiplication <- matrix1 * matrix2
> print("Multiplication:")
[1] "Multiplication:"
> print(matrix_multiplication)
     [,1] [,2] [,3]
[1,]  700 1600 2700
[2,] 4000 5500 7200
> # Division of matrices
> matrix_division <- matrix1 / matrix2
> print("Division:")
[1] "Division:"
> print(matrix_division)
          [,1]      [,2]      [,3]
[1,] 0.1428571 0.2500000 0.3333333
[2,] 0.4000000 0.4545455 0.5000000
```

4.Write a R program to create a matrix from a list of given vectors.

```
my_list <- list(vector1 = c(1, 2, 3),
                vector2 = c(4, 5, 6),
                vector3 = c(7, 8, 9))

my_matrix <- do.call(rbind, my_list)

print(my_matrix)
```

```
> my_list <- list(vector1 = c(1, 2, 3),
+                 vector2 = c(4, 5, 6),
+                 vector3 = c(7, 8, 9))
> my_matrix <- do.call(rbind, my_list)
> print(my_matrix)
        [,1] [,2] [,3]
vector1    1    2    3
vector2    4    5    6
vector3    7    8    9
>
```

5.Write a R program to find row and column index of maximum and minimum value in a given matrix.

```
# Define the matrix

my_matrix <- matrix(c(5, 10, 15, 20, 25, 30, 35, 40, 45), nrow =
3, ncol = 3, byrow = TRUE)

# Find the indices of the maximum and minimum values

max_value <- max(my_matrix)

min_value <- min(my_matrix)

max_indices <- which(my_matrix == max_value, arr.ind = TRUE)
```

```
min_indices <- which(my_matrix == min_value, arr.ind = TRUE)

# Extract the row and column indices

max_row <- max_indices[1, 1]

max_col <- max_indices[1, 2]

min_row <- min_indices[1, 1]

min_col <- min_indices[1, 2]

print(my_matrix)

# Print the results

print("Maximum Value:")

print(max_value)

print("Row Index of Maximum Value:")

print(max_row)

print("Column Index of Maximum Value:")

print(max_col)

print("Minimum Value:")

print(min_value)

print("Row Index of Minimum Value:")

print(min_row)

print("Column Index of Minimum Value:")

print(min_col)
```

```
> # Define the matrix
> my_matrix <- matrix(c(5, 10, 15, 20, 25, 30, 35, 40, 45), nrow = 3, ncol = 3, byrow = TRUE)
> # Find the indices of the maximum and minimum values
> max_value <- max(my_matrix)
> min_value <- min(my_matrix)
> max_indices <- which(my_matrix == max_value, arr.ind = TRUE)
> min_indices <- which(my_matrix == min_value, arr.ind = TRUE)
> # Extract the row and column indices
> max_row <- max_indices[1, 1]
> max_col <- max_indices[1, 2]
> min_row <- min_indices[1, 1]
> min_col <- min_indices[1, 2]
> print(my_matrix)
     [,1] [,2] [,3]
[1,]    5   10   15
[2,]   20   25   30
[3,]   35   40   45
```

```
> # Print the results
> print("Maximum Value:")
[1] "Maximum Value:"
> print(max_value)
[1] 45
> print("Row Index of Maximum Value:")
[1] "Row Index of Maximum Value:"
> print(max_row)
row
  3
> print("Column Index of Maximum Value:")
[1] "Column Index of Maximum Value:"
> print(max_col)
col
  3
> print("Minimum Value:")
[1] "Minimum Value:"
> print(min_value)
[1] 5
> print("Row Index of Minimum Value:")
[1] "Row Index of Minimum Value:"
> print(min_row)
row
  1
```

```
> print("Column Index of Minimum Value:")
[1] "Column Index of Minimum Value:"
> print(min_col)
col
  1
>
```

Arrays

6.Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors.

```
# Define the vectors

vector1 <- 1:9

vector2 <- 10:18

# Create the array

my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))

# Print the array

print(my_array)
```

```
> # Define the vectors
> vector1 <- 1:9
> vector2 <- 10:18
> # Create the array
> my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))
> # Print the array
> print(my_array)
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

>
```

7.Write a R program to create an 3 dimensional array of 24 elements using the dim() function.

```
# Create the array with dimensions 2x3x4

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

# Print the array

print(my_array)
```

```
> # Create the array with dimensions 2x3x4
> my_array <- array(1:24, dim = c(2, 3, 4))
> # Print the array
> 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

>
```

8.Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

```
vector1 <- 1:9

vector2 <- 10:18

# Create the array

my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))

# Access specific elements

second_row_second_matrix <- my_array[2, , 2]

element_3rd_row_3rd_column_1st_matrix <- my_array[3, 3, 1]

# Print the results

print("Second row of the second matrix:")

print(second_row_second_matrix)

print("Element in the 3rd row and 3rd column of the 1st
matrix:")

print(element_3rd_row_3rd_column_1st_matrix)
```

```
> vector1 <- 1:9
> vector2 <- 10:18
> # Create the array
> my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))
> # Access specific elements
> second_row_second_matrix <- my_array[2, , 2]
> element_3rd_row_3rd_column_1st_matrix <- my_array[3, 3, 1]
> # Print the results
> print("Second row of the second matrix:")
[1] "Second row of the second matrix:"
> print(second_row_second_matrix)
[1] 11 14 17
> print("Element in the 3rd row and 3rd column of the 1st matrix:")
[1] "Element in the 3rd row and 3rd column of the 1st matrix:"
> print(element_3rd_row_3rd_column_1st_matrix)
[1] 9
>
```

----------X----------

Thank you!