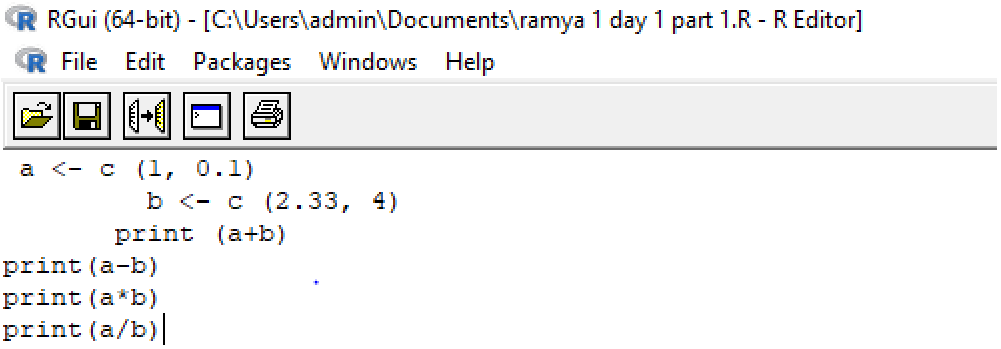
name: v.ramya sri
reg no:192124039

lab exercise 1(part1)

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

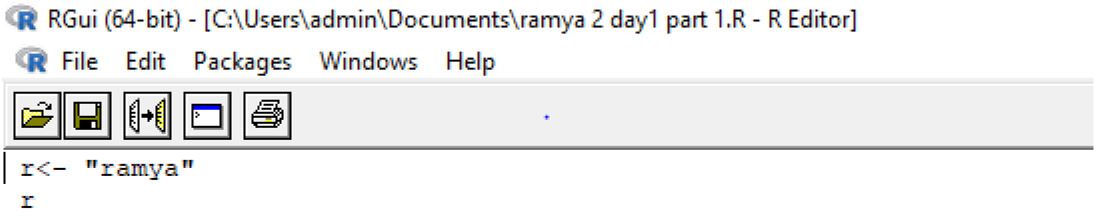
```
 a <- c (1, 0.1)
>	b <- c (2.33, 4)
>	print (a+b)
[1] 3.33 4.10
> print(a-b)
[1] -1.33 -3.90
> print(a*b)
[1] 2.33 0.40
> print(a/b)
[1] 0.4291845 0.0250000
>
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 1 day 1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
 a <- c (1, 0.1)
         b <- c (2.33, 4)
       print (a+b)
print(a-b)
print(a*b)
print(a/b)
```

2. Display a String on R Console.

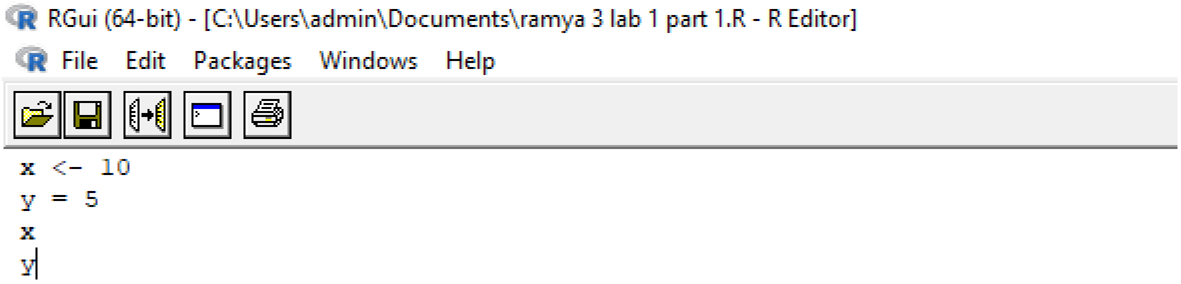
```
> r<- "ramya"
> r
[1] "ramya"
>
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 2 day1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
r<- "ramya"
r
```

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

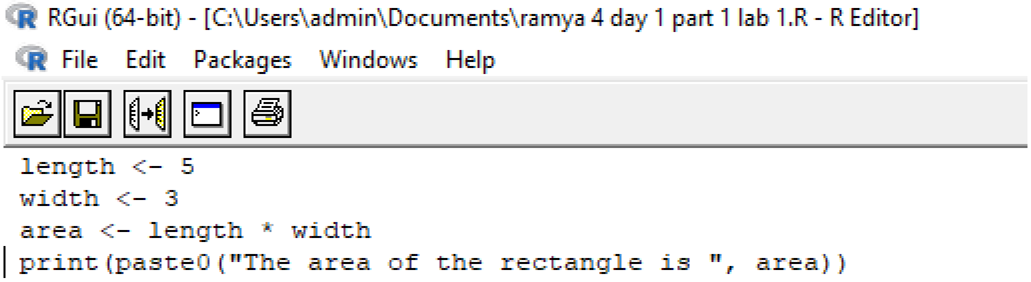
```
> x <- 10
> y = 5
> x
[1] 10
> y
[1] 5
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 3 lab 1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

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

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

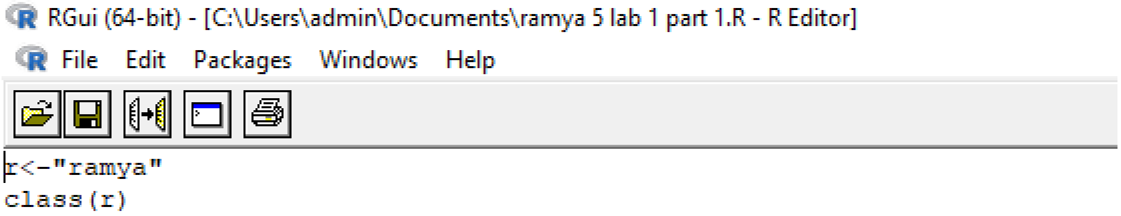
```
> length <- 5
> width <- 3
> area <- length * width
> print(paste0("The area of the rectangle is ", area))
[1] "The area of the rectangle is 15"
>
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 4 day 1 part 1 lab 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
length <- 5
width <- 3
area <- length * width
print(paste0("The area of the rectangle is ", area))
```

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

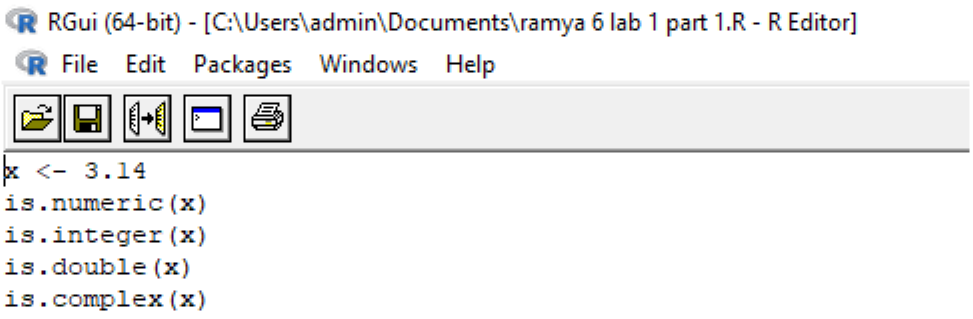
```
> r<-"ramya"
> class(r)
[1] "character"
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 5 lab 1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
r<-"ramya"
class(r)
```

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

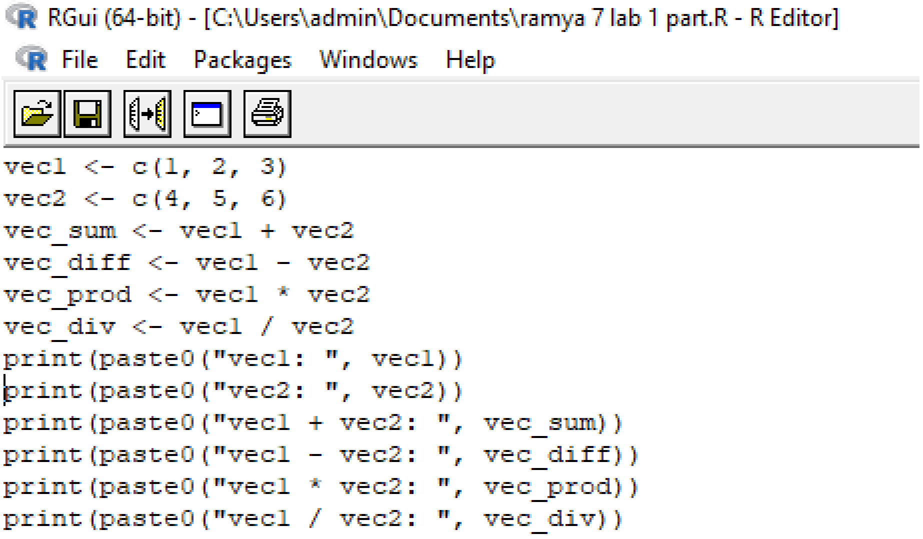
```
> x <- 3.14
> is.numeric(x)
[1] TRUE
> is.integer(x)
[1] FALSE
> is.double(x)
[1] TRUE
> is.complex(x)
[1] FALSE
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 6 lab 1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
x <- 3.14
is.numeric(x)
is.integer(x)
is.double(x)
is.complex(x)
```

7. Illustration of Vector Arithmetic.

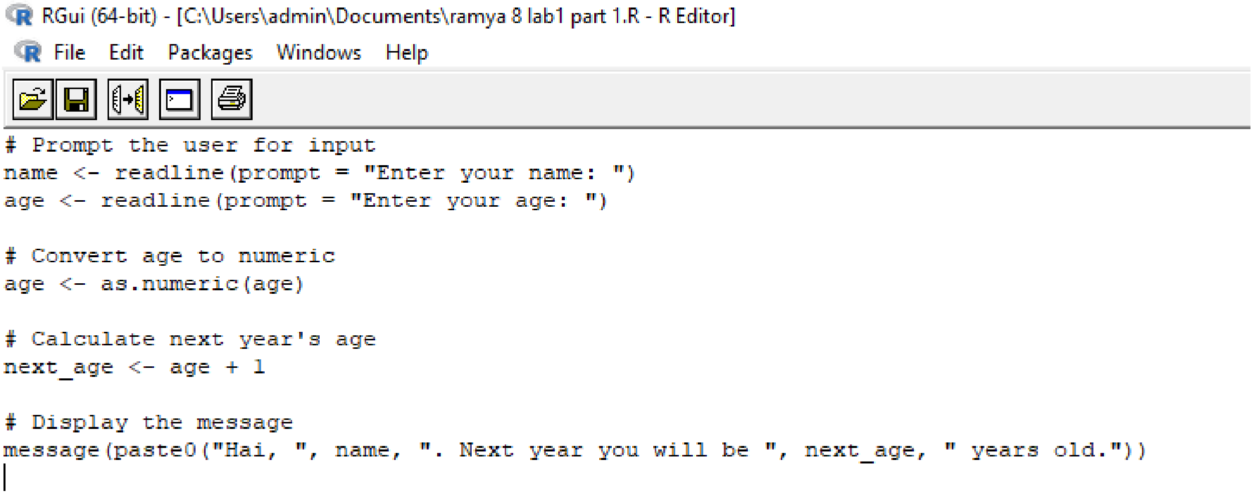
```
> vec1 <- c(1, 2, 3)
> vec2 <- c(4, 5, 6)
> vec_sum <- vec1 + vec2
> vec_diff <- vec1 - vec2
>
> vec_prod <- vec1 * vec2
> vec_div <- vec1 / vec2
> print(paste0("vec1: ", vec1))
[1] "vec1: 1" "vec1: 2" "vec1: 3"
> print(paste0("vec2: ", vec2))
[1] "vec2: 4" "vec2: 5" "vec2: 6"
> print(paste0("vec1 + vec2: ", vec_sum))
[1] "vec1 + vec2: 5" "vec1 + vec2: 7" "vec1 + vec2: 9"
> print(paste0("vec1 - vec2: ", vec_diff))
[1] "vec1 - vec2: -3" "vec1 - vec2: -3" "vec1 - vec2: -3"
> print(paste0("vec1 * vec2: ", vec_prod))
[1] "vec1 * vec2: 4"  "vec1 * vec2: 10" "vec1 * vec2: 18"
> print(paste0("vec1 / vec2: ", vec_div))
[1] "vec1 / vec2: 0.25" "vec1 / vec2: 0.4"  "vec1 / vec2: 0.5"
>
```


RGui (64-bit) - [C:\Users\admin\Documents\ramya 7 lab 1 part.R - R Editor]
File
Edit
Packages
Windows
Help

```
vec1 <- c(1, 2, 3)
vec2 <- c(4, 5, 6)
vec_sum <- vec1 + vec2
vec_diff <- vec1 - vec2
vec_prod <- vec1 * vec2
vec_div <- vec1 / vec2
print(paste0("vec1: ", vec1))
print(paste0("vec2: ", vec2))
print(paste0("vec1 + vec2: ", vec_sum))
print(paste0("vec1 - vec2: ", vec_diff))
print(paste0("vec1 * vec2: ", vec_prod))
print(paste0("vec1 / vec2: ", vec_div))
```

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”

```
> name <- readline(prompt = "Enter your name: ")
Enter your name: age <- readline(prompt = "Enter your age: ")
> age <- as.numeric(age)
> next_age <- age + 1
> message(paste0("Hai, ", name, ". Next year you will be ", next_age, " years old."))
Hai, age <- readline(prompt = "Enter your age: "). Next year you will be NA years old.
```

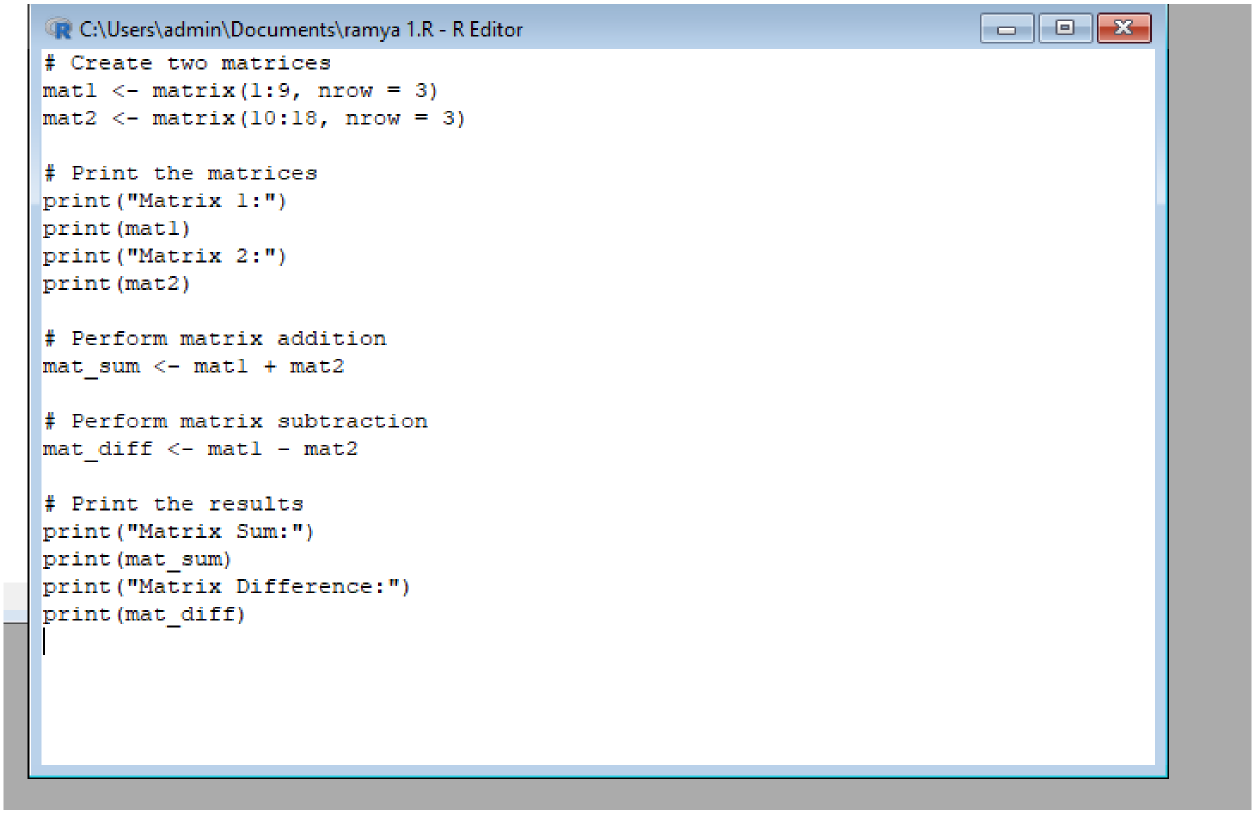

RGui (64-bit) - [C:\Users\admin\Documents\ramya 8 lab1 part 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
# Prompt the user for input
name <- readline(prompt = "Enter your name: ")
age <- readline(prompt = "Enter your age: ")

# Convert age to numeric
age <- as.numeric(age)

# Calculate next year's age
next_age <- age + 1

# Display the message
message(paste0("Hai, ", name, ". Next year you will be ", next_age, " years old."))
```

Part 2

1) Perform Matrix Addition &amp; Subtraction in R

```
> mat1 <- matrix(1:9, nrow = 3)
> mat2 <- matrix(10:18, nrow = 3)
> print("Matrix 1:")
[1] "Matrix 1:"
> print(mat1)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> print("Matrix 2:")
[1] "Matrix 2:"
> print(mat2)
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
>
> mat_sum <- mat1 + mat2
> mat_diff <- mat1 - mat2
>
> print("Matrix Sum:")
[1] "Matrix Sum:"
> print(mat_sum)
     [,1] [,2] [,3]
[1,]   11   17   23
[2,]   13   19   25
[3,]   15   21   27
> print("Matrix Difference:")
[1] "Matrix Difference:"
> print(mat_diff)
     [,1] [,2] [,3]
[1,]   -9   -9   -9
```

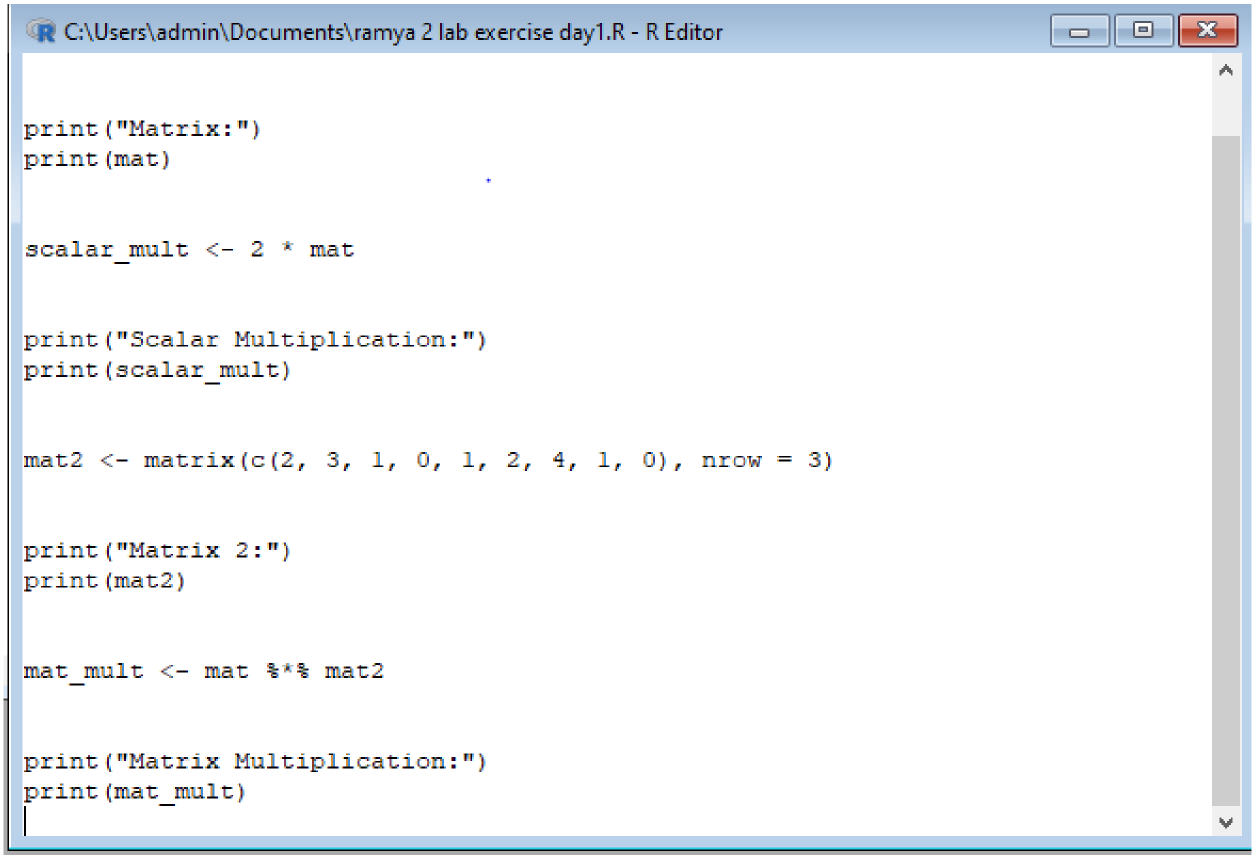


```
# Create two matrices
mat1 <- matrix(1:9, nrow = 3)
mat2 <- matrix(10:18, nrow = 3)

# Print the matrices
print("Matrix 1:")
print(mat1)
print("Matrix 2:")
print(mat2)

# Perform matrix addition
mat_sum <- mat1 + mat2

# Perform matrix subtraction
mat_diff <- mat1 - mat2

# Print the results
print("Matrix Sum:")
print(mat_sum)
print("Matrix Difference:")
print(mat_diff)
```

2) Perform Scalar multiplication and matrix multiplication in R

```
> mat2 <- matrix(c(2, 3, 1, 0, 1, 2, 4, 1, 0), nrow = 3)
>
>
> print("Matrix 2:")
[1] "Matrix 2:"
> print(mat2)
     [,1] [,2] [,3]
[1,]    2    0    4
[2,]    3    1    1
[3,]    1    2    0
>
>
> mat_mult <- mat %*% mat2
>
>
> print("Matrix Multiplication:")
[1] "Matrix Multiplication:"
> print(mat_mult)
```

```
     [,1] [,2] [,3]
[1,]   21   18    8
[2,]   27   21   13
[3,]   33   24   18
```



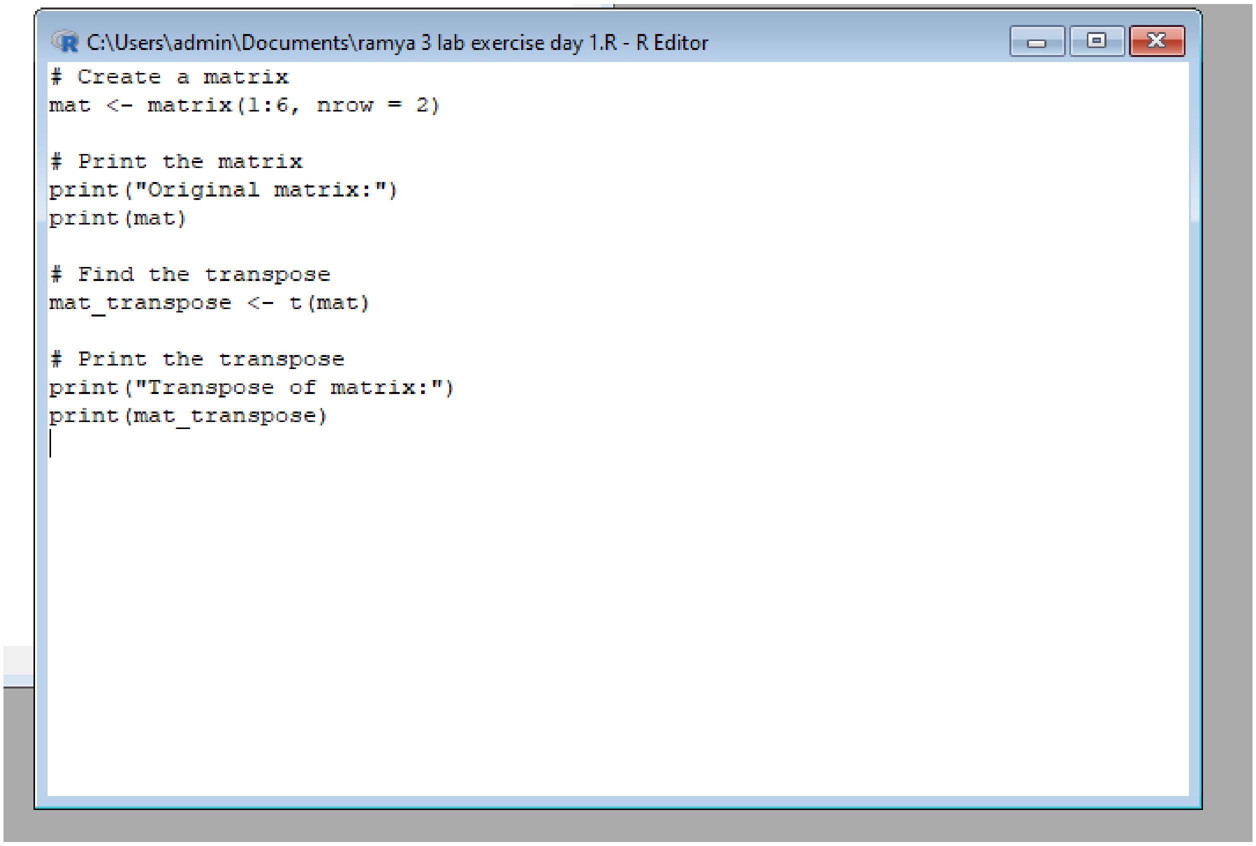
C:\Users\admin\Documents\ramya 2 lab exercise day1.R - R Editor

```
print("Matrix:")
print(mat)


scalar_mult <- 2 * mat


print("Scalar Multiplication:")
print(scalar_mult)


mat2 <- matrix(c(2, 3, 1, 0, 1, 2, 4, 1, 0), nrow = 3)


print("Matrix 2:")
print(mat2)


mat_mult <- mat %*% mat2


print("Matrix Multiplication:")
print(mat_mult)
```

3) Find Transpose of matrix in R.

```
> # Create a matrix
> mat <- matrix(1:6, nrow = 2)
>
> # Print the matrix
> print("Original matrix:")
[1] "Original matrix:"
> print(mat)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> # Find the transpose
> mat_transpose <- t(mat)
```

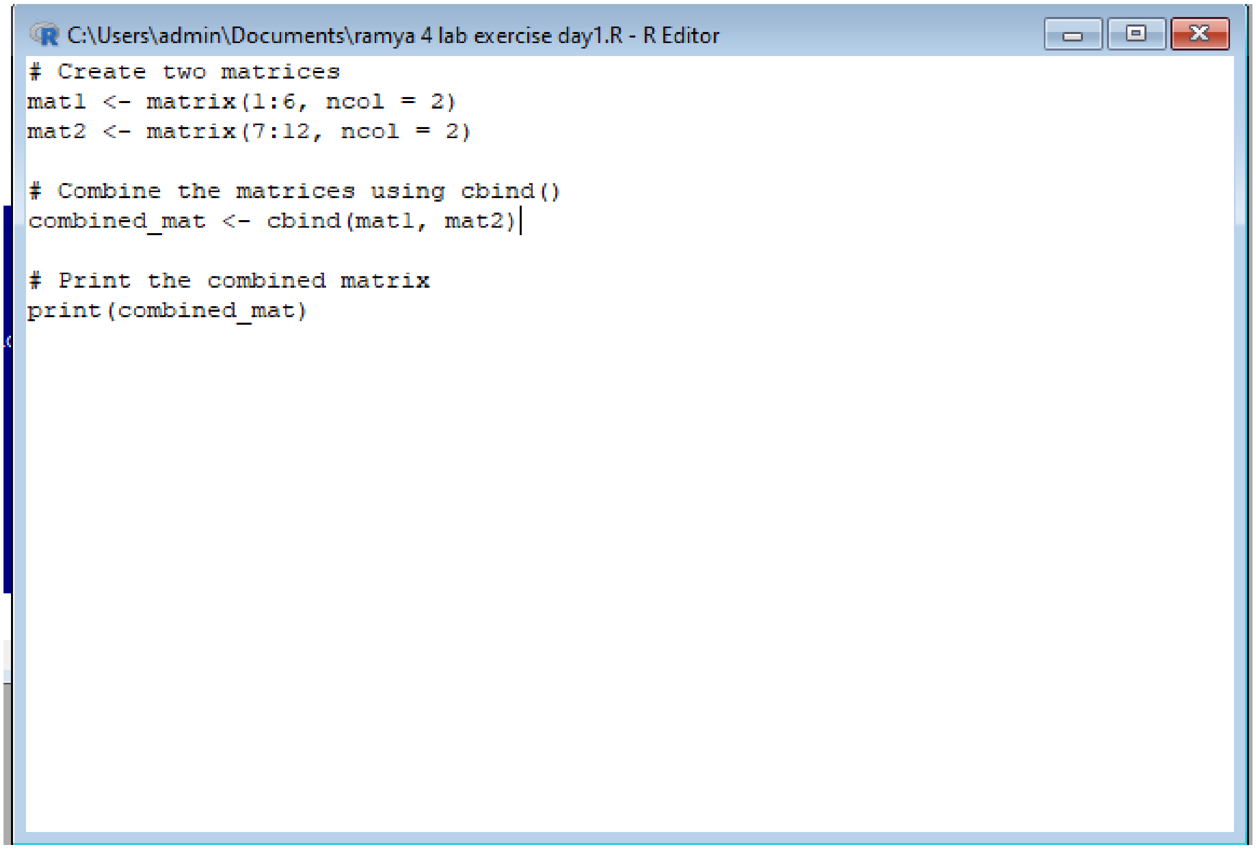
```
>
> # Print the transpose
> print("Transpose of matrix:")
[1] "Transpose of matrix:"
> print(mat_transpose)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> print(mat_transpose)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```


C:\Users\admin\Documents\ramya 3 lab exercise day 1.R - R Editor

```
# Create a matrix
mat <- matrix(1:6, nrow = 2)

# Print the matrix
print("Original matrix:")
print(mat)

# Find the transpose
mat_transpose <- t(mat)

# Print the transpose
print("Transpose of matrix:")
print(mat_transpose)
```

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

```
> # Create two matrices
> mat1 <- matrix(1:6, ncol = 2)
> mat2 <- matrix(7:12, ncol = 2)
>
> # Combine the matrices using cbind()
> combined_mat <- cbind(mat1, mat2)
>
> # Print the combined matrix
> print(combined_mat)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
```

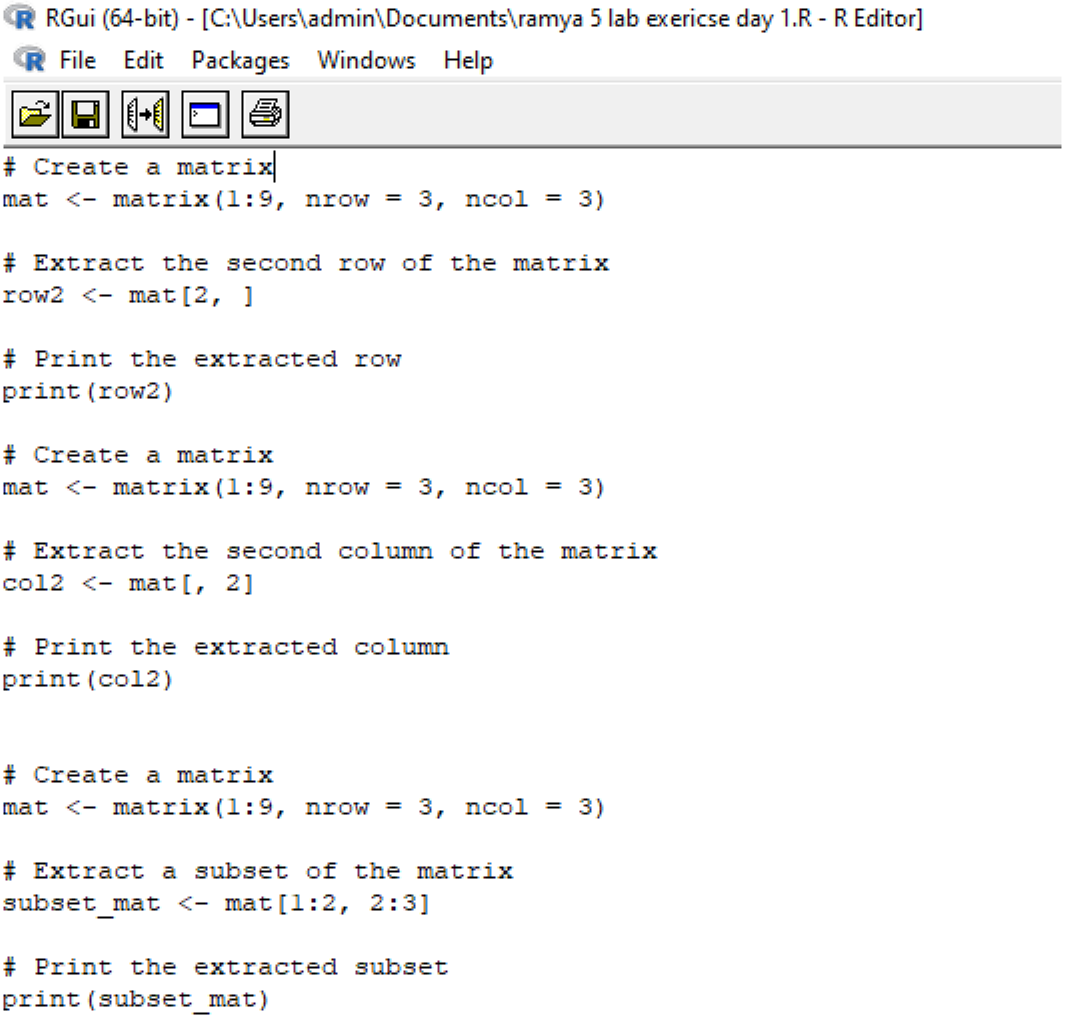

C:\Users\admin\Documents\ramya 4 lab exercise day1.R - R Editor

```
# Create two matrices
mat1 <- matrix(1:6, ncol = 2)
mat2 <- matrix(7:12, ncol = 2)

# Combine the matrices using cbind()
combined_mat <- cbind(mat1, mat2)

# Print the combined matrix
print(combined_mat)
```

5) Deconstruct a matrix in R

```
# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract the second row of the matrix
row2 <- mat[2, ]
```

```
# Print the extracted row
print(row2)

# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract the second column of the matrix
col2 <- mat[, 2]

# Print the extracted column
print(col2)


# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract a subset of the matrix
subset_mat <- mat[1:2, 2:3]

# Print the extracted subset
print(subset_mat)
```



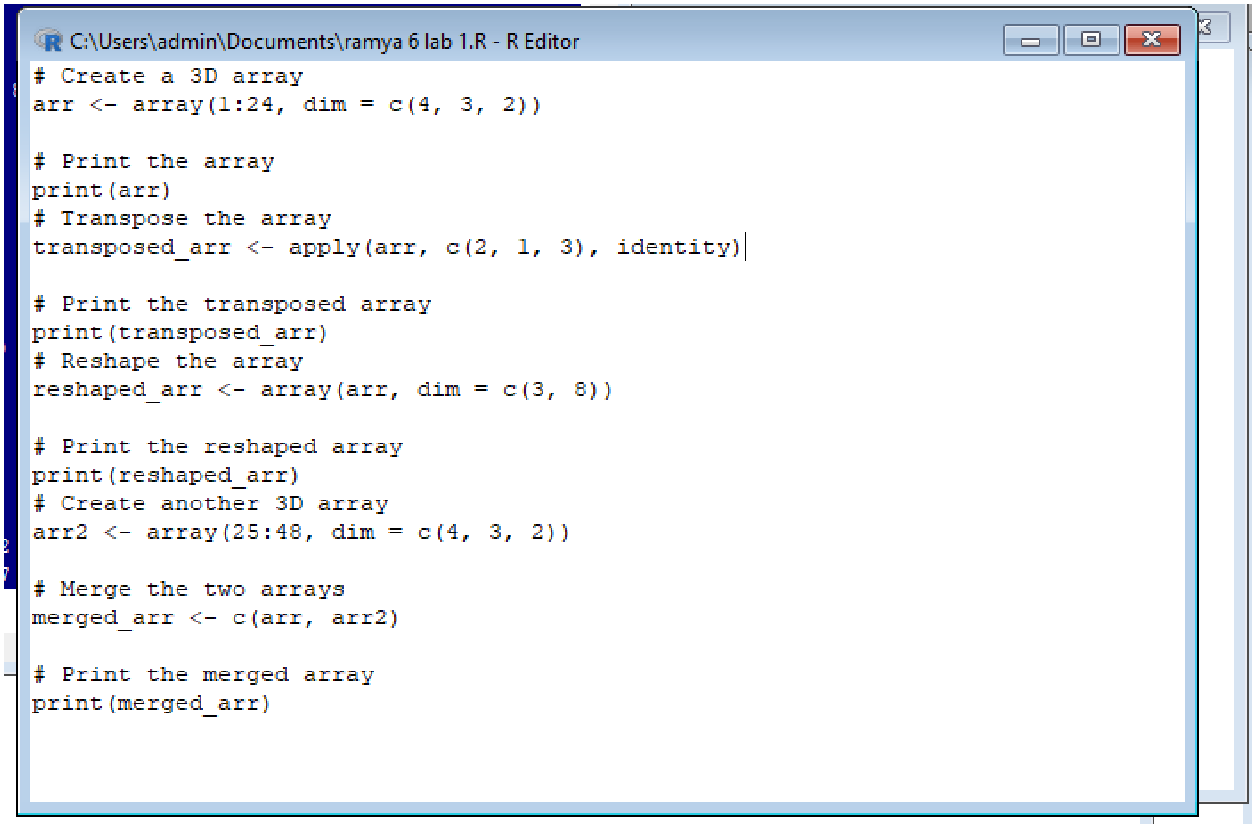
RGui (64-bit) - [C:\Users\admin\Documents\ramya 5 lab exericse day 1.R - R Editor]
File
Edit
Packages
Windows
Help

```
# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract the second row of the matrix
row2 <- mat[2, ]

# Print the extracted row
print(row2)

# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract the second column of the matrix
col2 <- mat[, 2]

# Print the extracted column
print(col2)


# Create a matrix
mat <- matrix(1:9, nrow = 3, ncol = 3)

# Extract a subset of the matrix
subset_mat <- mat[1:2, 2:3]

# Print the extracted subset
print(subset_mat)
```

6) Perform array manipulation in R .

```
> # Transpose the array
> transposed_arr <- apply(arr, c(2, 1, 3), identity)
>
> # Print the transposed array
> print(transposed_arr)
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

```
, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   14   15   16
[2,]   17   18   19   20
[3,]   21   22   23   24

> # Reshape the array
> reshaped_arr <- array(arr, dim = c(3, 8))
>
> # Print the reshaped array
> print(reshaped_arr)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    4    7   10   13   16   19   22
[2,]    2    5    8   11   14   17   20   23
[3,]    3    6    9   12   15   18   21   24
> # Create another 3D array
> arr2 <- array(25:48, dim = c(4, 3, 2))
>
> # Merge the two arrays
> merged_arr <- c(arr, arr2)
>
> # Print the merged array
> print(merged_arr)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
>
```

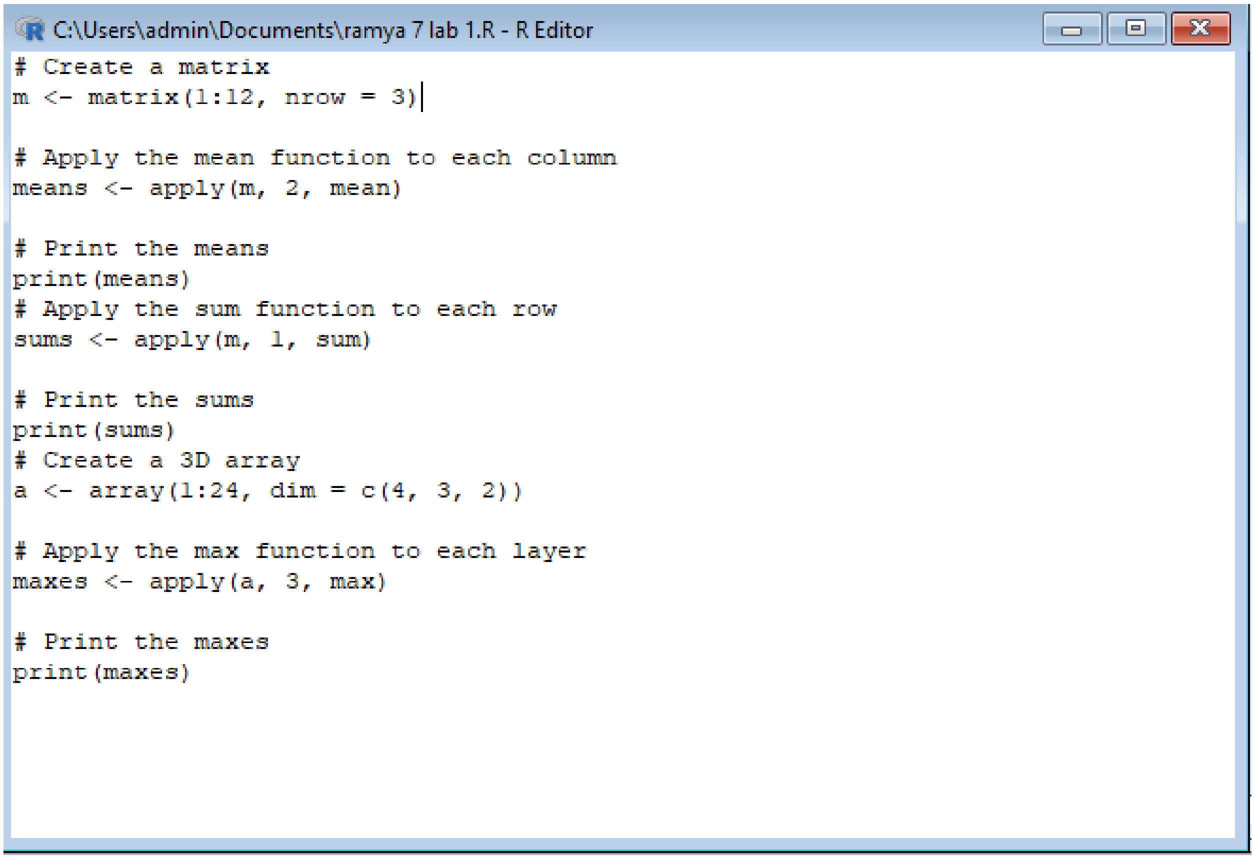


```
# Create a 3D array
arr <- array(1:24, dim = c(4, 3, 2))

# Print the array
print(arr)
# Transpose the array
transposed_arr <- apply(arr, c(2, 1, 3), identity)

# Print the transposed array
print(transposed_arr)
# Reshape the array
reshaped_arr <- array(arr, dim = c(3, 8))

# Print the reshaped array
print(reshaped_arr)
# Create another 3D array
arr2 <- array(25:48, dim = c(4, 3, 2))

# Merge the two arrays
merged_arr <- c(arr, arr2)

# Print the merged array
print(merged_arr)
```

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

```
> # Create a matrix
> m <- matrix(1:12, nrow = 3)
>
> # Apply the mean function to each column
> means <- apply(m, 2, mean)
>
> # Print the means
> print(means)
[1]  2  5  8 11
> # Apply the sum function to each row
> sums <- apply(m, 1, sum)
Error in if (n <= 1) { : the condition has length > 1
>
> # Print the sums
> print(sums)
Error in print(sums) : object 'sums' not found
> # Create a 3D array
> a <- array(1:24, dim = c(4, 3, 2))
>
```

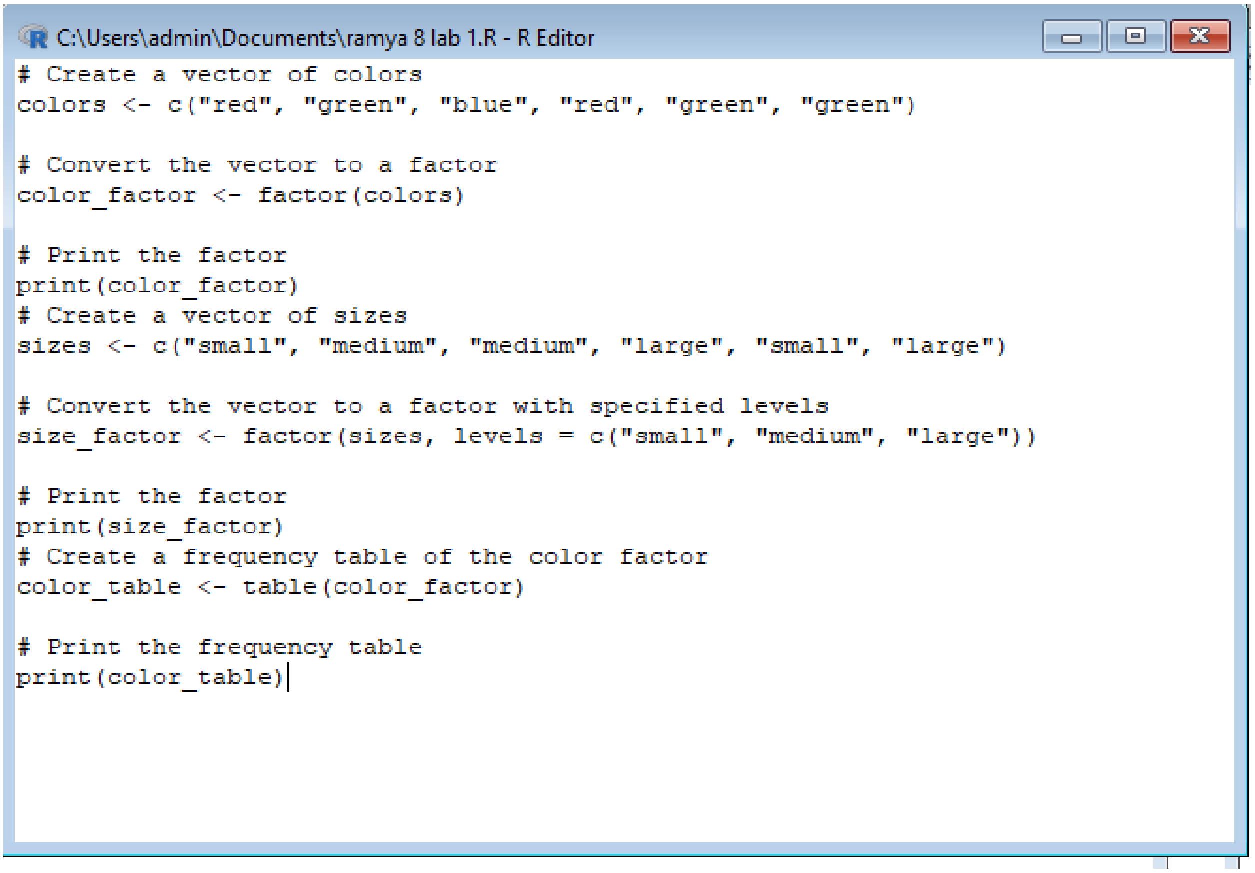
```
> # Apply the max function to each layer
> maxes <- apply(a, 3, max)
>
> # Print the maxes
> print(maxes)
[1] 12 24
>
```



```
# Create a matrix
m <- matrix(1:12, nrow = 3)

# Apply the mean function to each column
means <- apply(m, 2, mean)

# Print the means
print(means)
# Apply the sum function to each row
sums <- apply(m, 1, sum)

# Print the sums
print(sums)
# Create a 3D array
a <- array(1:24, dim = c(4, 3, 2))

# Apply the max function to each layer
maxes <- apply(a, 3, max)

# Print the maxes
print(maxes)
```

8) Demonstrate Factor data structure in R.

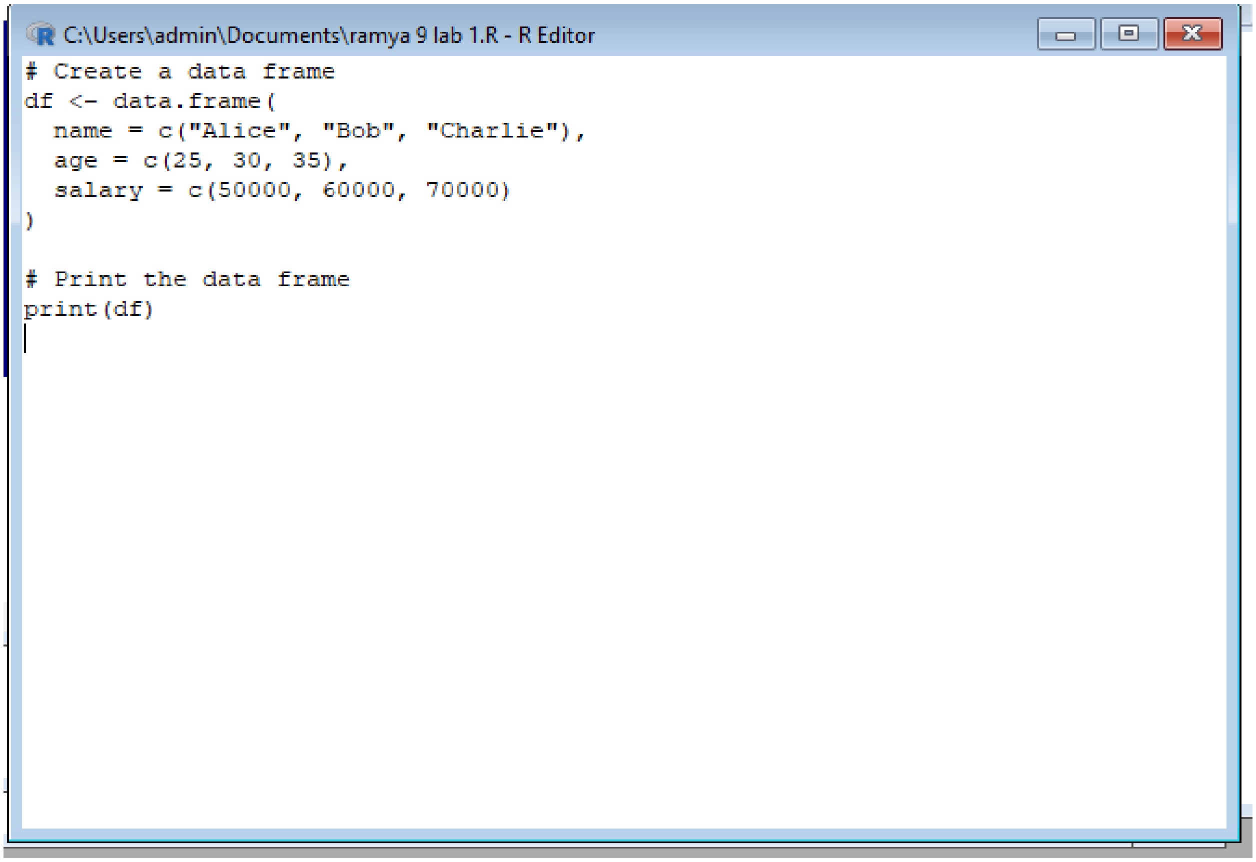
```
> # Create a vector of colors
> colors <- c("red", "green", "blue", "red", "green", "green")
>
> # Convert the vector to a factor
> color_factor <- factor(colors)
>
> # Print the factor
> print(color_factor)
[1] red   green blue  red   green green
Levels: blue green red
> # Create a vector of sizes
> sizes <- c("small", "medium", "medium", "large", "small", "large")
>
> # Convert the vector to a factor with specified levels
> size_factor <- factor(sizes, levels = c("small", "medium", "large"))
>
> # Print the factor
> print(size_factor)
[1] small  medium medium large  small  large
Levels: small medium large
> # Create a frequency table of the color factor
> color_table <- table(color_factor)
>
> # Print the frequency table
> print(color_table)
color_factor
 blue green   red
    1     3     2
>
```



```
# Create a vector of colors
colors <- c("red", "green", "blue", "red", "green", "green")

# Convert the vector to a factor
color_factor <- factor(colors)

# Print the factor
print(color_factor)
# Create a vector of sizes
sizes <- c("small", "medium", "medium", "large", "small", "large")

# Convert the vector to a factor with specified levels
size_factor <- factor(sizes, levels = c("small", "medium", "large"))

# Print the factor
print(size_factor)
# Create a frequency table of the color factor
color_table <- table(color_factor)

# Print the frequency table
print(color_table)
```

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

```
> # Create a data frame
> df <- data.frame(
+   name = c("Alice", "Bob", "Charlie"),
+   age = c(25, 30, 35),
+   salary = c(50000, 60000, 70000)
+ )
>
> # Print the data frame
> print(df)
     name age salary
1   Alice  25  50000
2     Bob  30  60000
3 Charlie  35  70000
```

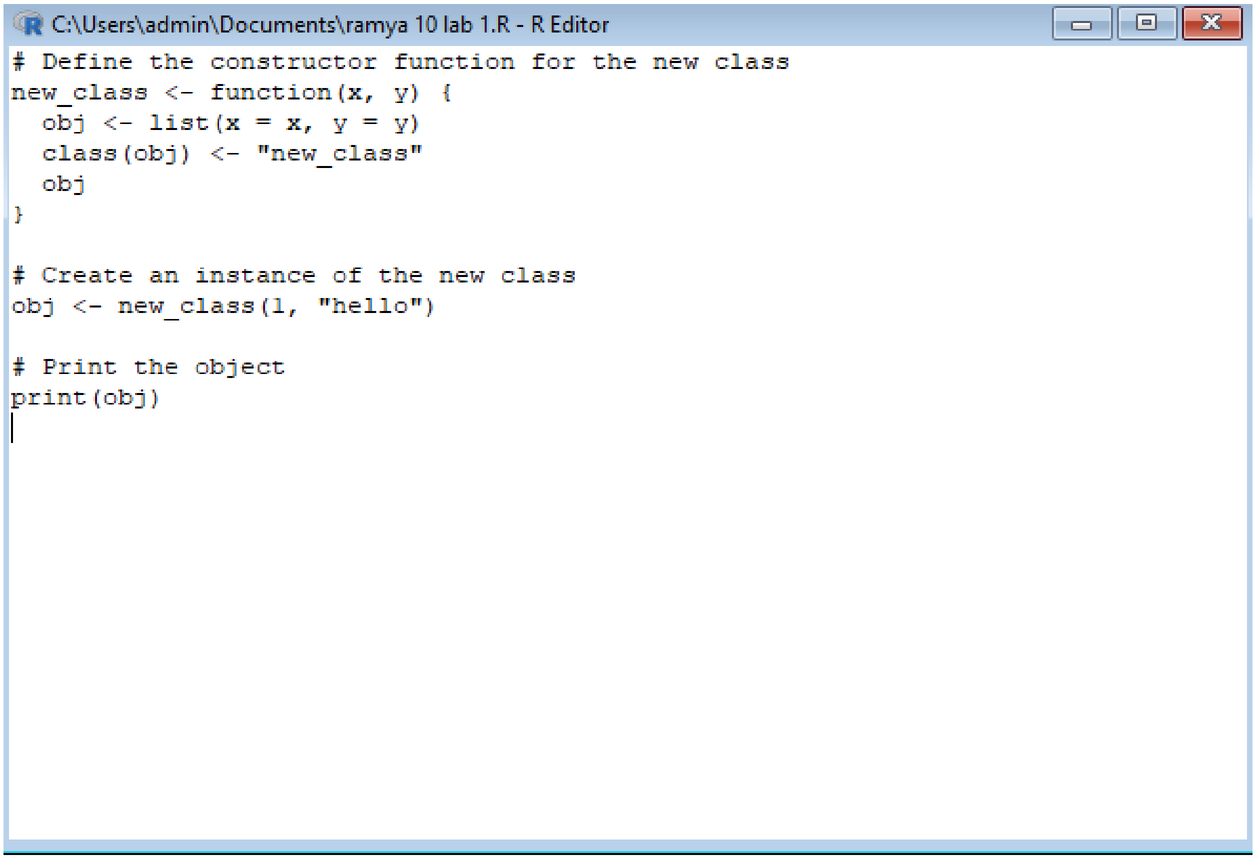


```
# Create a data frame
df <- data.frame(
  name = c("Alice", "Bob", "Charlie"),
  age = c(25, 30, 35),
  salary = c(50000, 60000, 70000)
)

# Print the data frame
print(df)
```

10) Demonstrate the creation of S3 class in R.

```
> # Define the constructor function for the new class
> new_class <- function(x, y) {
+   obj <- list(x = x, y = y)
+   class(obj) <- "new_class"
+   obj
+ }
```

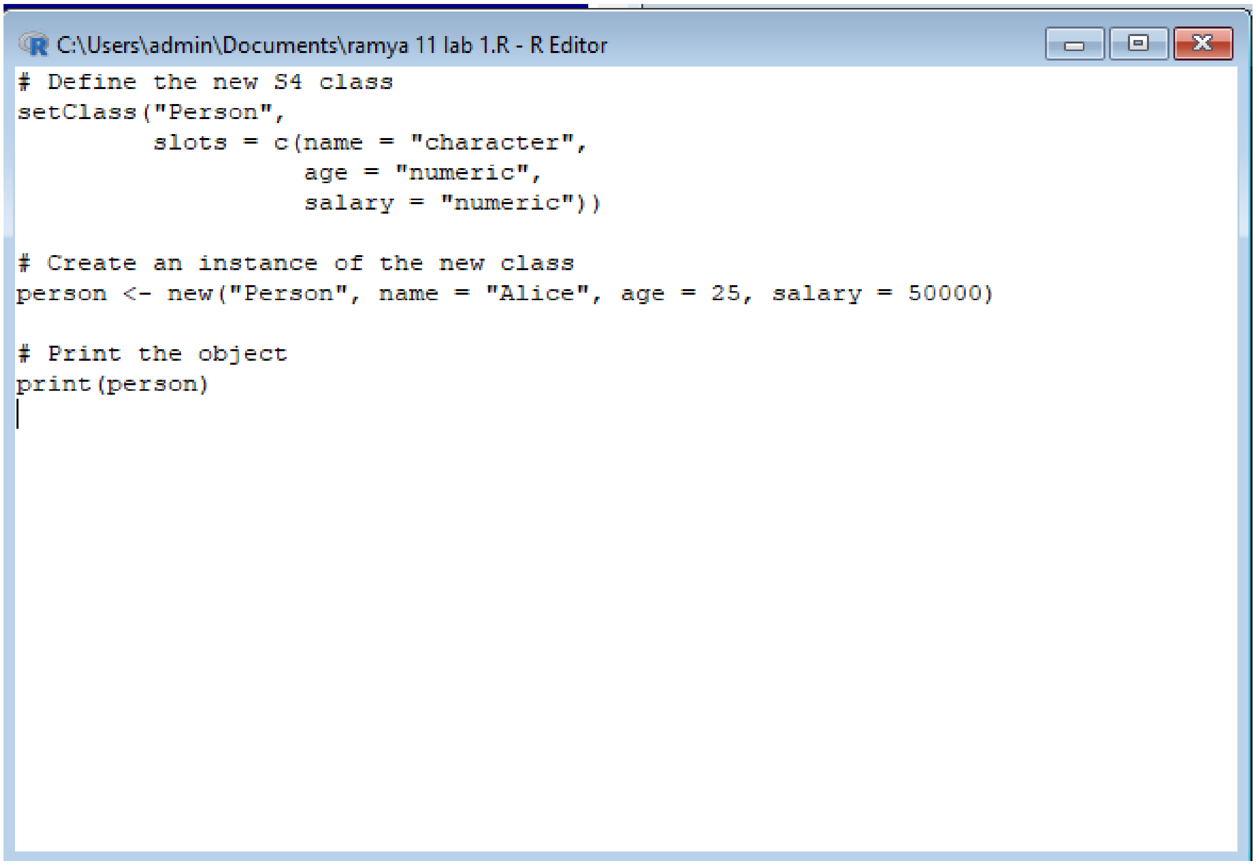
```
>
> # Create an instance of the new class
> obj <- new_class(1, "hello")
>
> # Print the object
> print(obj)
$x
[1] 1

$y
[1] "hello"

attr(,"class")
[1] "new_class"
>
```



```
# Define the constructor function for the new class
new_class <- function(x, y) {
  obj <- list(x = x, y = y)
  class(obj) <- "new_class"
  obj
}

# Create an instance of the new class
obj <- new_class(1, "hello")

# Print the object
print(obj)
```

11) Demonstrate the creation of S4 class in R.

```
> # Define the new S4 class
> setClass("Person",
+         slots = c(name = "character",
+                   age = "numeric",
+                   salary = "numeric"))
>
> # Create an instance of the new class
> person <- new("Person", name = "Alice", age = 25, salary = 50000)
>
> # Print the object
> print(person)
An object of class "Person"
Slot "name":
[1] "Alice"

Slot "age":
[1] 25

Slot "salary":
[1] 50000
```

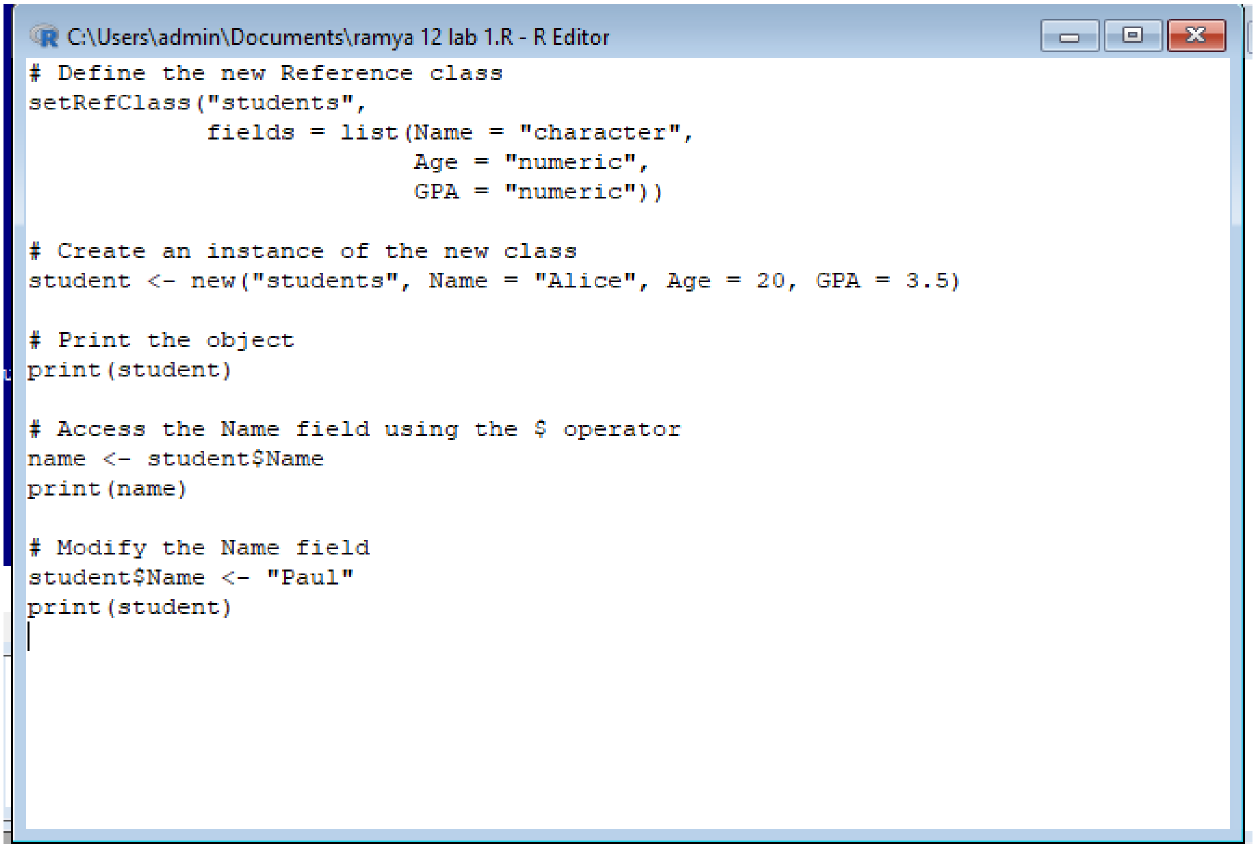


```
# Define the new S4 class
setClass("Person",
         slots = c(name = "character",
                   age = "numeric",
                   salary = "numeric"))

# Create an instance of the new class
person <- new("Person", name = "Alice", age = 25, salary = 50000)

# Print the object
print(person)
```

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.

```
> # Create an instance of the new class
> student <- new("students", Name = "Alice", Age = 20, GPA = 3.5)
>
> # Print the object
> print(student)
Reference class object of class "students"
Field "Name":
[1] "Alice"
Field "Age":
[1] 20
Field "GPA":
[1] 3.5
>
> # Access the Name field using the $ operator
> name <- student$Name
> print(name)
[1] "Alice"
>
> # Modify the Name field
> student$Name <- "Paul"
> print(student)
Reference class object of class "students"
Field "Name":
[1] "Paul"
Field "Age":
[1] 20
Field "GPA":
[1] 3.5
>
```



```
# Define the new Reference class
setRefClass("students",
            fields = list(Name = "character",
                          Age = "numeric",
                          GPA = "numeric"))

# Create an instance of the new class
student <- new("students", Name = "Alice", Age = 20, GPA = 3.5)

# Print the object
print(student)

# Access the Name field using the $ operator
name <- student$Name
print(name)

# Modify the Name field
student$Name <- "Paul"
print(student)
```