ITA0448~ R PROGRAMMING

S.DHANUSH KUMAR

192121154

ASSESSMENT 1 DAY 5

1.Write a R program to Create the following details

a. x= sample(-50:50, 10, replace=TRUE).and print the value of x

PROGRAM:

```
x <- sample(-50:50, 10, replace = TRUE)

print(x)
```

OUTPUT:

```
> x <- sample(-50:50, 10, replace = TRUE)
> print(x)
 [1]  14   7 -31 -19   6  -1   7  28   9   6
>

>
```

b. To create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to

50 and sum of numbers from 20 to 50.

PROGRAM:

```
numbers <- 20:50

mean_numbers <- mean(numbers)

sum_numbers <- sum(numbers)

print(paste("Mean of numbers from 20 to 50:",
mean_numbers))

print(paste("Sum of numbers from 20 to 50:",
sum_numbers))
```

OUTPUT:

```
> numbers <- 20:50
> mean_numbers <- mean(numbers)
> sum_numbers <- sum(numbers)
> print(paste("Mean of numbers from 20 to 50:", mean
_numbers))
[1] "Mean of numbers from 20 to 50: 35"
> print(paste("Sum of numbers from 20 to 50:", sum_n
umbers))
[1] "Sum of numbers from 20 to 50: 1085"
>

>
```

2. To create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two

vectors.vector1 = c(1,3,4,5) and vector2 = c(10,11,12,13,14,15)

a. Print vector1, vector2

b. Print new array

PROGRAM:

```
# Define the vectors
vector1 <- c(1, 3, 4, 5)
vector2 <- c(10, 11, 12, 13, 14, 15)

# Create the two 3x3 matrices from the vectors
matrix1 <- matrix(vector1, nrow = 3, ncol = 3)
matrix2 <- matrix(vector2, nrow = 3, ncol = 3)

# Combine the matrices into an array
array_2_matrices <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

# Print the vectors, matrices, and array
print(paste("Vector 1:", vector1))
print(paste("Vector 2:", vector2))
print("Matrix 1:")
print(matrix1)
print("Matrix 2:")
print(matrix2)
print("Array of 2 matrices:")
print(array_2_matrices)
```

OUTPUT:

```
> # Define the vectors
> vector1 <- c(1, 3, 4, 5)
> vector2 <- c(10, 11, 12, 13, 14, 15)
>
> # Create the two 3x3 matrices from the vectors
> matrix1 <- matrix(vector1, nrow = 3, ncol = 3)
> matrix2 <- matrix(vector2, nrow = 3, ncol = 3)
>
> # Combine the matrices into an array
> array_2_matrices <- array(c(matrix1, matrix2), dim
= c(3, 3, 2))
>
> # Print the vectors, matrices, and array
> print(paste("Vector 1:", vector1))
[1] "Vector 1: 1" "Vector 1: 3" "Vector 1: 4" "Vector 1:
5"
> print(paste("Vector 2:", vector2))
[1] "Vector 2: 10" "Vector 2: 11" "Vector 2: 12" "Vect
or 2: 13"
[5] "Vector 2: 14" "Vector 2: 15"
> print("Matrix 1:")
[1] "Matrix 1:"
> print(matrix1)
     [,1] [,2] [,3]
```

```
[1,]    1    5    4
[2,]    3    1    5
[3,]    4    3    1
> print("Matrix 2:")
[1] "Matrix 2:"
> print(matrix2)
     [,1] [,2] [,3]
[1,]   10   13   10
[2,]   11   14   11
[3,]   12   15   12
> print("Array of 2 matrices:")
[1] "Array of 2 matrices:"
> print(array_2_matrices)
, , 1

     [,1] [,2] [,3]
[1,]    1    5    4
[2,]    3    1    5
[3,]    4    3    1

, , 2

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

>

>
```

3. Write a R program to merge two given lists into one list. n1 = list (1,2,3) c1 = list(&quot;Raja&quot;,

**&quot;Rani&quot;, &quot;Prince&quot;)
i) Write a R program to convert a given list to vector.n1 = list (1,2,3)c1 = list(4,5,6)**

PROGRAM:

```
# Define the list
n1 <- list(1, 2, 3)

# Convert the list to a vector
vector_n1 <- unlist(n1)

# Print the vector
print(vector_n1)
```

OUTPUT:

```
> # Define the list
> n1 <- list(1, 2, 3)
>
> # Convert the list to a vector
> vector_n1 <- unlist(n1)
>
> # Print the vector
> print(vector_n1)
[1] 1 2 3
>
```

4. Consider A=matrix(c(2,0,1,3),ncol=2) and B=matrix(c(5,2,4,-1), ncol=2).

a) Find A + B b) Find A – B c) Find A * B d) Find 3A + 3B

PROGRAM:

```
# Define the matrices
A <- matrix(c(2, 0, 1, 3), ncol = 2)
B <- matrix(c(5, 2, 4, -1), ncol = 2)

# a) Find A + B
add_AB <- A + B
print("A + B:")
print(add_AB)

# b) Find A - B
sub_AB <- A - B
print("A - B:")
print(sub_AB)

# c) Find A * B
mult_AB <- A %*% B
print("A * B:")
print(mult_AB)
```

```
# d) Find 3A + 3B
scalar_mult_A <- 3 * A
scalar_mult_B <- 3 * B
add_scalar_mult_AB <- scalar_mult_A + scalar_mult_B
print("3A + 3B:")
print(add_scalar_mult_AB)
```

OUTPUT:

```
> # Define the matrices
> A <- matrix(c(2, 0, 1, 3), ncol = 2)
> B <- matrix(c(5, 2, 4, -1), ncol = 2)
>
> # a) Find A + B
> add_AB <- A + B
> print("A + B:")
[1] "A + B:"
> print(add_AB)
     [,1] [,2]
[1,]    7    5
[2,]    2    2
>
> # b) Find A - B
> sub_AB <- A - B
> print("A - B:")
[1] "A - B:"
> print(sub_AB)
     [,1] [,2]
[1,]   -3   -3
[2,]   -2    4
>
```

```
> # c) Find A * B
> mult_AB <- A %*% B
> print("A * B:")
[1] "A * B:"
> print(mult_AB)
     [,1] [,2]
[1,]   12    7
[2,]    6   -3
>
> # d) Find 3A + 3B
> scalar_mult_A <- 3 * A
> scalar_mult_B <- 3 * B
> add_scalar_mult_AB <- scalar_mult_A + scalar_mult_B
> print("3A + 3B:")
[1] "3A + 3B:"
> print(add_scalar_mult_AB)
     [,1] [,2]
[1,]   21   15
[2,]    6    6
>

>
```

5. Write a nested loop, where the outer for() loop increments “a” 3 times, and the inner for() loop increments “b” 3 times. The break statement exits the inner for() loop after 2 incrementations. The nested loop prints the values of variables, “a” and “b“.

PROGRAM:

```
# Define variables

a <- 1

b <- 1
```

```
# Nested for loop
for (i in 1:3) {
  for (j in 1:3) {
    # Print values of a and b
    cat("a:", a, "b:", b, "\n")

    # Break inner loop after 2 incrementations
    if (j == 2) {
      break
    }

    # Increment b
    b <- b + 1
  }

  # Increment a
  a <- a + 1
}
```

OUTPUT:

```
> # Define variables
> a <- 1
> b <- 1
>
```

```
> # Nested for loop
> for (i in 1:3) {
+     for (j in 1:3) {
+         # Print values of a and b
+         cat("a:", a, "b:", b, "\n")
+
+         # Break inner loop after 2 incrementations
+         if (j == 2) {
+             break
+         }
+
+         # Increment b
+         b <- b + 1
+     }
+
+     # Increment a
+     a <- a + 1
+ }
a: 1 b: 1
a: 1 b: 2
a: 2 b: 2
a: 2 b: 3
a: 3 b: 3
a: 3 b: 4
>

>
```

6. (a) Suppose we have a fruit basket with 20 apples. Store the number of apples in a variable

my_apples.

(b) Every tasty fruit basket needs oranges, so we decide to add six oranges. As a data analyst , the

reflex is to immediately create a variable my_oranges and assign the value 6 to it. Next , calculate how

many pieces of fruit we have in total in the variable my_fruit.

PROGRAM:

```
# (a) Store the number of apples in a variable

my_apples <- 20


# (b) Add 6 oranges and calculate total number of fruits

my_oranges <- 6

my_fruit <- my_apples + my_oranges


# Print the number of apples, oranges, and total fruit

cat("Number of apples:", my_

apples, "\n")

cat("Number of oranges:", my_oranges, "\n")

cat("Total number of fruit:", my_fruit, "\n")
```

OUTPUT:

```
> # (a) Store the number of apples in a variable
> my_apples <- 20
>
> # (b) Add 6 oranges and calculate total number of fruits
> my_oranges <- 6
> my_fruit <- my_apples + my_oranges
>
> # Print the number of apples, oranges, and total fruit
> cat("Number of apples:", my_apples, "\n")
Number of apples: 20
> cat("Number of oranges:", my_oranges, "\n")
Number of oranges: 6
```

```
> cat("Total number of fruit:", my_fruit, "\n")
Total number of fruit: 26
>

>
```

7. Perform the following operations using R:
a. Initialize 3 character variables named age,employed and salary.
b. Transform age to numeric type and store in the variable age_clean.
c. Initialize employed_clean with the result obtained by converting employed to logical type.
d. Convert the respondent's salary to a numeric and store it in the variable salary_clean.

PROGRAM:

```
# (a) Initialize 3 character variables

age <- "30"

employed <- "yes"

salary <- "$50,000"


# (b) Transform age to numeric type

age_clean <- as.numeric(age)


# (c) Convert employed to logical type

employed_clean <- as.logical(employed)


# (d) Convert salary to numeric type
# First, remove the "$" and "," characters
salary_clean <- gsub("\\$|\\,", "", salary)

# Convert to numeric type
salary_clean <- as.numeric(salary_clean)

# Print the cleaned variables
cat("age_clean:", age_clean, "\n")
cat("employed_clean:", employed_clean, "\n")
cat("salary_clean:", salary_clean, "\n")
```

OUTPUT:

```
> # (a) Initialize 3 character variables
> age <- "30"
> employed <- "yes"
> salary <- "$50,000"
>
> # (b) Transform age to numeric type
> age_clean <- as.numeric(age)
>
> # (c) Convert employed to logical type
> employed_clean <- as.logical(employed)
>
> # (d) Convert salary to numeric type
> # First, remove the "$" and "," characters
> salary_clean <- gsub("\\$|\\,", "", salary)
>
> # Convert to numeric type
> salary_clean <- as.numeric(salary_clean)
>
> # Print the cleaned variables
> cat("age_clean:", age_clean, "\n")
age_clean: 30
> cat("employed_clean:", employed_clean, "\n")
employed_clean: NA
> cat("salary_clean:", salary_clean, "\n")
salary_clean: 50000
```

8. Create the following vectors in R.

a = (5,10, 15, 20, ..., 160)

b = (87, 86, 85, ..., 56)

Use vector arithmetic to multiply these vectors and call the result d. Select subsets of d to identify the

following.

(a) What are the 19th, 20th, and 21st elements of d?

(b) What are all of the elements of d which are less than 2000?

(c) How many elements of d are greater than 6000?

PROGRAM:

```
# Create vector a using seq() function
a <- seq(from=5, to=160, by=5)

# Create vector b using colon operator
b <- 87:56

# Multiply vectors a and b to get vector d
d <- a * b

# (a) Subset d to get 19th, 20th, and 21st elements
cat("19th, 20th, and 21st elements of d: ", d[19:21], "\n")
```

```
# (b) Subset d to get elements less than 2000

d_less_than_2000 <- d[d < 2000]

cat("Elements of d less than 2000: ", d_less_than_2000,
"\n")


# (c) Count elements of d greater than 6000

d_greater_than_6000 <- d[d > 6000]

cat("Number of elements of d greater than 6000: ",
length(d_greater_than_6000), "\n")
```

OUTPUT:

```
> # Create vector a using seq() function
> a <- seq(from=5, to=160, by=5)
>
> # Create vector b using colon operator
> b <- 87:56
>
> # Multiply vectors a and b to get vector d
> d <- a * b
>
> # (a) Subset d to get 19th, 20th, and 21st elements
> cat("19th, 20th, and 21st elements of d: ", d[19:21], "\n")
19th, 20th, and 21st elements of d:  6555 6800 7035
>
> # (b) Subset d to get elements less than 2000
> d_less_than_2000 <- d[d < 2000]
> cat("Elements of d less than 2000: ", d_less_than_2000, "\n")
Elements of d less than 2000:  435 860 1275 1680
>
> # (c) Count elements of d greater than 6000
> d_greater_than_6000 <- d[d > 6000]
> cat("Number of elements of d greater than 6000: ", length(d_greater_than_6000), "\n")
Number of elements of d greater than 6000:  16
```

9. You have an employee data-set, which comprises of two columns-&gt;”name” and designation”, add

a third column which would indicate the current date and time.

This is the employee data-set:

PROGRAM:

# Load the employee data-set into a data frame

employee_data <- read.csv("employee_data.csv")

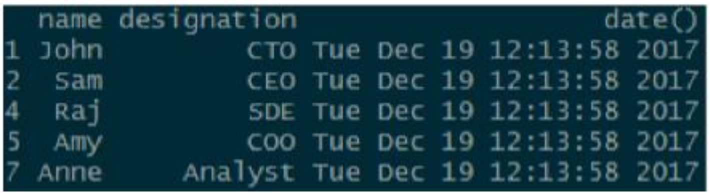
# Add a new column for current date and time

employee_data$current_date_and_time <- Sys.time()

# Save the updated data-set to a new CSV file

write.csv(employee_data, "employee_data_with_datetime.csv", row.names = FALSE)

OUTPUT:



```
  name designation                    date()
1 John         CTO Tue Dec 19 12:13:58 2017
2  Sam         CEO Tue Dec 19 12:13:58 2017
4  Raj         SDE Tue Dec 19 12:13:58 2017
5  Amy         COO Tue Dec 19 12:13:58 2017
7 Anne     Analyst Tue Dec 19 12:13:58 2017
```

10. Implement a multiplication game. A while loop that gives the user two random numbers from 2 to

12 and asks the user to multiply them. Only exit the loop after five correct answers. Try using

as.integer(readline())

PROGRAM:

```
correct_answers <- 0

while(correct_answers < 5) {
  # Generate two random numbers between 2 and 12
  num1 <- sample(2:12, 1)
  num2 <- sample(2:12, 1)

  # Ask the user to multiply the numbers
  cat("What is", num1, "x", num2, "? ")
  user_answer <- as.integer(readline())

  # Check if the user's answer is correct
  if(user_answer == num1 * num2) {
    correct_answers <- correct_answers + 1
    cat("Correct!\n\n")
  } else {
    cat("Sorry, that's incorrect. The answer was", num1 * num2, "\n\n")
  }
}

cat("Congratulations, you got 5 correct answers!")
```

OUTPUT:

```
> correct_answers <- 0
>
> while(correct_answers < 5) {
```

```
+    # Generate two random numbers between 2 and 12
+    num1 <- sample(2:12, 1)
+    num2 <- sample(2:12, 1)
+
+    # Ask the user to multiply the numbers
+    cat("What is", num1, "x", num2, "? ")
+    user_answer <- as.integer(readline())
+
+    # Check if the user's answer is correct
+    if(user_answer == num1 * num2) {
+       correct_answers <- correct_answers + 1
+       cat("Correct!\n\n")
+    } else {
+       cat("Sorry, that's incorrect. The answer was", num1 * num2, "\n\n")
+    }
+ }
What is 2 x 4 ?
```

11. Create a Attendance sheet of the course “R Programming”.All are present for the course and total strength of the students is 30. There are 15 male students register number from 191611258 to 191611272 and 15 female students of Register number from 191611273 to 191611287. Use data frames to create the Attendance Sheet.(Refer the Sample attendance sheet for 6 students is given below)

S ample Attendance Sheet

```
regno gender attendance
1 191611258 MALE PRESENT
2 191611259 MALE PRESENT
3 191611260 MALE PRESENT
4 191611261 FEMALE PRESENT
5 191611262 FEMALE PRESENT
6 191611263 FEMALE PRESENTK
```

PROGRAM:

```
# Create a data frame with the register numbers of all the students
regno <- c(seq(191611258, 191611272), seq(191611273, 191611287))
attendance_df <- data.frame(regno)


# Add a column to indicate the gender of each student
attendance_df$gender <- c(rep("MALE", 15), rep("FEMALE", 15))


# Add a column to indicate that all students are present
attendance_df$attendance <- "PRESENT"


# Print the attendance sheet
print(attendance_df)
```

OUTPUT:

```
> # Create a data frame with the register numbers of all the students
> regno <- c(seq(191611258, 191611272), seq(191611273, 191611287))
> attendance_df <- data.frame(regno)
>
> # Add a column to indicate the gender of each student
> attendance_df$gender <- c(rep("MALE", 15), rep("FEMALE", 15))
>
> # Add a column to indicate that all students are present
> attendance_df$attendance <- "PRESENT"
>
> # Print the attendance sheet
> print(attendance_df)
       regno gender attendance
1  191611258   MALE    PRESENT
2  191611259   MALE    PRESENT
3  191611260   MALE    PRESENT
4  191611261   MALE    PRESENT
5  191611262   MALE    PRESENT
6  191611263   MALE    PRESENT
```

```
7  191611264   MALE    PRESENT
8  191611265   MALE    PRESENT
9  191611266   MALE    PRESENT
10 191611267   MALE    PRESENT
11 191611268   MALE    PRESENT
12 191611269   MALE    PRESENT
13 191611270   MALE    PRESENT
14 191611271   MALE    PRESENT
15 191611272   MALE    PRESENT
16 191611273 FEMALE    PRESENT
17 191611274 FEMALE    PRESENT
18 191611275 FEMALE    PRESENT
19 191611276 FEMALE    PRESENT
20 191611277 FEMALE    PRESENT
21 191611278 FEMALE    PRESENT
22 191611279 FEMALE    PRESENT
23 191611280 FEMALE    PRESENT
24 191611281 FEMALE    PRESENT
25 191611282 FEMALE    PRESENT
26 191611283 FEMALE    PRESENT
27 191611284 FEMALE    PRESENT
28 191611285 FEMALE    PRESENT
29 191611286 FEMALE    PRESENT
30 191611287 FEMALE    PRESENT
>
```

12. Create two vectors named v and w with the following contents:

v :21,55,84,12,13,15

w : 9,44,22,33,14,35

A) Print the length of the vectors B) Print all elements of the vectors

C) Print the sum of the elements in each vector. D)Find the mean of each vector. (Use R&#39;s mean() function)

E) Add vectors v and w. F) Multiply vectors v and w.

G) In vector v select all elements that are greater than 2.

H) In vector w select all elements that are less than 20.

PROGRAM:

# Create vectors v and w

```r
v <- c(21, 55, 84, 12, 13, 15)
w <- c(9, 44, 22, 33, 14, 35)

# Print the length of the vectors
cat("Length of vector v:", length(v), "\n")
cat("Length of vector w:", length(w), "\n\n")

# Print all elements of the vectors
cat("Elements of vector v:", v, "\n")
cat("Elements of vector w:", w, "\n\n")

# Print the sum of the elements in each vector
cat("Sum of vector v:", sum(v), "\n")
cat("Sum of vector w:", sum(w), "\n\n")

# Find the mean of each vector
cat("Mean of vector v:", mean(v), "\n")
cat("Mean of vector w:", mean(w), "\n\n")

# Add vectors v and w
cat("v + w:", v + w, "\n\n")

# Multiply vectors v and w
cat("v * w:", v * w, "\n\n")
```

```
# Select all elements in v that are greater than 2
cat("Elements in v greater than 2:", v[v > 2], "\n\n")

# Select all elements in w that are less than 20
cat("Elements in w less than 20:", w[w < 20], "\n\n")
```

OUTPUT:

```
> # Create vectors v and w
> v <- c(21, 55, 84, 12, 13, 15)
> w <- c(9, 44, 22, 33, 14, 35)
>
> # Print the length of the vectors
> cat("Length of vector v:", length(v), "\n")
Length of vector v: 6
> cat("Length of vector w:", length(w), "\n\n")
Length of vector w: 6

>
> # Print all elements of the vectors
> cat("Elements of vector v:", v, "\n")
Elements of vector v: 21 55 84 12 13 15
> cat("Elements of vector w:", w, "\n\n")
Elements of vector w: 9 44 22 33 14 35

>
> # Print the sum of the elements in each vector
> cat("Sum of vector v:", sum(v), "\n")
Sum of vector v: 200
> cat("Sum of vector w:", sum(w), "\n\n")
Sum of vector w: 157

>
> # Find the mean of each vector
> cat("Mean of vector v:", mean(v), "\n")
Mean of vector v: 33.33333
> cat("Mean of vector w:", mean(w), "\n\n")
Mean of vector w: 26.16667

>
> # Add vectors v and w
> cat("v + w:", v + w, "\n\n")
v + w: 30 99 106 45 27 50

>
> # Multiply vectors v and w
> cat("v * w:", v * w, "\n\n")
v * w: 189 2420 1848 396 182 525
```

```
>
> # Select all elements in v that are greater than 2
> cat("Elements in v greater than 2:", v[v > 2], "\n\n")
Elements in v greater than 2: 21 55 84 12 13 15

>
> # Select all elements in w that are less than 20
> cat("Elements in w less than 20:", w[w < 20], "\n\n")
Elements in w less than 20: 9 14
```

13. lapply function is applied to all elements of the input and it returns a list and saaply function is

applied to all elements of the input and it returns a vector. Demonstrate the use of sapply and

lapply with the following vector.

movies&lt;-c(&quot;SPYDERMAN&quot;,&quot;BATMAN&quot;,&quot;VERTIGO&quot;,&quot;CHINATOWN&quot;)

Convert these elements of vector into lowercase letters.

PROGRAM:

```
# Create the vector

movies <- c("SPYDERMAN", "BATMAN", "VERTIGO",
"CHINATOWN")


# Using lapply to convert the elements to lowercase and
return a list

lowercase_l <- lapply(movies, tolower)

cat("Output of lapply:\n")
```

```
print(lowercase_l)


# Using sapply to convert the elements to lowercase and return a vector

lowercase_s <- sapply(movies, tolower)

cat("\nOutput of sapply:\n")

print(lowercase_s)
```

OUTPUT:

```
> # Create the vector
> movies <- c("SPYDERMAN", "BATMAN", "VERTIGO", "CHINATOWN")
>
> # Using lapply to convert the elements to lowercase and return a list
> lowercase_l <- lapply(movies, tolower)
> cat("Output of lapply:\n")
Output of lapply:
> print(lowercase_l)
[[1]]
[1] "spyderman"

[[2]]
[1] "batman"

[[3]]
[1] "vertigo"

[[4]]
[1] "chinatown"

>
> # Using sapply to convert the elements to lowercase and return a vector
> lowercase_s <- sapply(movies, tolower)
> cat("\nOutput of sapply:\n")

Output of sapply:
> print(lowercase_s)
  SPYDERMAN      BATMAN     VERTIGO   CHINATOWN
"spyderman"    "batman"   "vertigo" "chinatown"
```

14. Create dataframe dataframe1 with the following vectors,

Mark1=c(35,45,67)

Mark2=c(56,89,99)

Mark3=c(78,75,83)

Use sapply and lapply function to find minimum marks ,maximum mark and average of all marks

PROGRAM:

```
# Create the data frame
dataframe1 <- data.frame(
  Mark1 = c(35, 45, 67),
  Mark2 = c(56, 89, 99),
  Mark3 = c(78, 75, 83)
)

# Using lapply to find minimum, maximum and average of all marks and return a list
min_max_avg_l <- lapply(dataframe1, function(x) c(min(x), max(x), mean(x)))
cat("Output of lapply:\n")
print(min_max_avg_l)

# Using sapply to find minimum, maximum and average of all marks and return a matrix
```

```
min_max_avg_s <- sapply(dataframe1, function(x) c(min(x),
max(x), mean(x)))

cat("\nOutput of sapply:\n")

print(min_max_avg_s)
```

OUTPUT:

```
> # Create the data frame
> dataframe1 <- data.frame(
+     Mark1 = c(35, 45, 67),
+     Mark2 = c(56, 89, 99),
+     Mark3 = c(78, 75, 83)
+ )
> 
> # Using lapply to find minimum, maximum and average of all marks and return a list
> min_max_avg_l <- lapply(dataframe1, function(x) c(min(x), max(x), mean(x)))
> cat("Output of lapply:\n")
Output of lapply:
> print(min_max_avg_l)
$Mark1
[1] 35 67 49

$Mark2
[1] 56.00000 99.00000 81.33333

$Mark3
[1] 75.00000 83.00000 78.66667

> 
> # Using sapply to find minimum, maximum and average of all marks and return a matrix
> min_max_avg_s <- sapply(dataframe1, function(x) c(min(x), max(x), mean(x)))
> cat("\nOutput of sapply:\n")

Output of sapply:
> print(min_max_avg_s)
     Mark1    Mark2    Mark3
[1,]    35 56.00000 75.00000
[2,]    67 99.00000 83.00000
[3,]    49 81.33333 78.66667
```

15. Write a R Program :

a. To find the multiplication table (from 1 to 10)

b. To find factorial of number

c. To check if the input number is odd or even

d. To check if the input number is prime or not

e. To find sum of natural numbers up-to 10, without formula using loop statement

PROGRAM:

a. To find the multiplication table (from 1 to 10)

```
# Using nested for loops to print multiplication table
for(i in 1:10){
  cat(paste("Multiplication table of", i, ":\n"))
  for(j in 1:10){
    cat(paste(i, "x", j, "=", i*j, "\n"))
  }
  cat("\n")
}
```

OUTPUT:

```
> # Using nested for loops to print multiplication table
> for(i in 1:10){
+     cat(paste("Multiplication table of", i, ":\n"))
+     for(j in 1:10){
+         cat(paste(i, "x", j, "=", i*j, "\n"))
+     }
+     cat("\n")
+ }
Multiplication table of 1 :
1 x 1 = 1
```

1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication table of 2 :
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Multiplication table of 3 :
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Multiplication table of 4 :
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

Multiplication table of 5 :
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Multiplication table of 6 :
6 x 1 = 6

6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

Multiplication table of 7 :
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

Multiplication table of 8 :
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80

Multiplication table of 9 :
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

Multiplication table of 10 :
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100

b. To find factorial of number

PROGRAM:

```
# Using a function to calculate factorial
factorial <- function(n){
  if(n==0){
    return(1)
  } else {
    return(n*factorial(n-1))
  }
}

# Example usage
cat("Factorial of 5 is", factorial(5))
```

OUTPUT:

```
> # Using a function to calculate factorial
> factorial <- function(n){
+     if(n==0){
+         return(1)
+     } else {
+         return(n*factorial(n-1))
+     }
+ }
> 
> # Example usage
> cat("Factorial of 5 is", factorial(5))
Factorial of 5 is 120>
```

c. To check if the input number is odd or even

PROGRAM:

```
# Using if else statement to check if a number is odd or even
check_even_odd <- function(n){
  if(n %% 2 == 0){
    return(paste(n, "is even."))
  } else {
    return(paste(n, "is odd."))
  }
}

# Example usage
cat(check_even_odd(7))
```

OUTPUT

```
> # Using if else statement to check if a number is odd or even
> check_even_odd <- function(n){
+     if(n %% 2 == 0){
+         return(paste(n, "is even."))
+     } else {
+         return(paste(n, "is odd."))
+     }
+ }
> 
> # Example usage
> cat(check_even_odd(7))
7 is odd.>
>
```

d. To check if the input number is prime or not

PROGRAM

```
# Using a function to check if a number is prime
is_prime <- function(n){
  if(n <= 1){
    return(FALSE)
  } else if(n == 2){
    return(TRUE)
  } else if(n %% 2 == 0){
    return(FALSE)
  } else {
    for(i in 3:ceiling(sqrt(n)), by=2){
      if(n %% i == 0){
        return(FALSE)
      }
    }
    return(TRUE)
  }
}

# Example usage
cat(is_prime(17))
```

OUTPUT

```
> # Using a function to check if a number is prime
> is_prime <- function(n){
+    if(n <= 1){
+        return(FALSE)
+    } else if(n == 2){
+        return(TRUE)
+    } else if(n %% 2 == 0){
```

```
+        return(FALSE)
+    } else {
+        for(i in 3:ceiling(sqrt(n)), by=2){
```

e. To find sum of natural numbers up-to 10, without formula using loop statement

PROGRAM:

```
# Using a for loop to find the sum of natural numbers up-to 10
sum_nat <- 0
for(i in 1:10){
  sum_nat <- sum_nat + i
}
cat("The sum of natural numbers up-to 10 is", sum_nat)
```

OUTPUT:

```
> # Using a for loop to find the sum of natural numbers up-to 10
> sum_nat <- 0
> for(i in 1:10){
+     sum_nat <- sum_nat + i
+ }
> cat("The sum of natural numbers up-to 10 is", sum_nat)
The sum of natural numbers up-to 10 is 55>
```

16. a. Create a data frame from four given vectors.

name =c (&#39;Anastasia&#39;, &#39;Dima&#39;, &#39;Katherine&#39;, &#39;James&#39;, &#39;Emily&#39;, &#39;Michael&#39;, &#39;Matthew&#39;, &#39;Laura’, &#39;Kevin&#39;,

&#39;Jonas&#39;)

score = c (12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)

attempts =c (1, 3, 2, 3, 2, 3, 1, 1, 2, 1)

qualify = c (&#39;yes&#39;, &#39;no&#39;, &#39;yes&#39;, &#39;no&#39;, &#39;no&#39;, &#39;yes&#39;, &#39;yes&#39;, &#39;no&#39;, &#39;no&#39;, &#39;yes&#39;)

b. Write a R program to extract first two rows from a given data frame.

c. Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame

d. Find the average score with respect to first, second, and third attempts. Don’t use any special in

build function for this task.

e. Write a R program to create a list containing a vector, a matrix and a list and give names to the

elements in the list. Access and print the first and second element of the list

PROGRAM:

. a. Create a data frame from four given vectors.

PROGRAM:

```
name <- c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas')

score <- c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)

attempts <- c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)

qualify <- c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
'yes')


dataframe <- data.frame(name, score, attempts, qualify)
```

OUTPUT:

```
> name <- c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jon
as')
> score <- c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)
> attempts <- c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)
> qualify <- c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
>
> dataframe <- data.frame(name, score, attempts, qualify)
```

b. Write a R program to extract first two rows from a given data frame.

PROGRAM:

```
first_two_rows <- dataframe[1:2, ]
```

OUTPUT:

```
> name <- c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jon
as')
> score <- c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)
> attempts <- c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)
> qualify <- c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
>
> dataframe <- data.frame(name, score, attempts, qualify)
```

```
>
> first_two_rows <- dataframe[1:2, ]
```

c. Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame

PROGRAM:

```
third_fifth_rows <- dataframe[c(3, 5), c(1, 3)]
```

OUTPUT:

```
> name <- c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jon
as')
> score <- c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)
> attempts <- c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)
> qualify <- c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
>
> dataframe <- data.frame(name, score, attempts, qualify)
>
> first_two_rows <- dataframe[1:2, ]
```

d. Find the average score with respect to first, second, and third attempts. Don’t use any special in

PROGRAM:

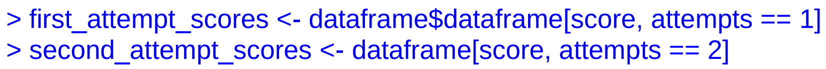
```
first_attempt_scores <- dataframe$dataframe[score,
attempts == 1]
second_attempt_scores <- dataframe[score, attempts == 2]
third_attempt_scores <- dataframe[score, attempts == 3]
```

```
mean_first_attempt_score <- sum(first_attempt_scores) / length(first_attempt_scores)

mean_second_attempt_score <- sum(second_attempt_scores) / length(second_attempt_scores)

mean_third_attempt_score <- sum(third_attempt_scores) / length(third_attempt_scores)
```

OUTPUT:



```
> first_attempt_scores <- dataframe$dataframe[score, attempts == 1]
> second_attempt_scores <- dataframe[score, attempts == 2]
```

e. Write a R program to create a list containing a vector, a matrix and a list and give names to the

elements in the list. Access and print the first and second element of the list

PROGRAM:

```
vector1 <- c(1, 2, 3)

matrix1 <- matrix(1:9, nrow = 3)

list1 <- list(a = 'apple', b = 'banana', c = 'cherry')


my_list <- list(vector1, matrix1, list1)

names(my_list) <- c('My Vector', 'My Matrix', 'My List')
```

```
print(my_list[[1]])
print(my_list[[2]])
```

OUTPUT:

```
> vector1 <- c(1, 2, 3)
> matrix1 <- matrix(1:9, nrow = 3)
> list1 <- list(a = 'apple', b = 'banana', c = 'cherry')
>
> my_list <- list(vector1, matrix1, list1)
> names(my_list) <- c('My Vector', 'My Matrix', 'My List')
>
> print(my_list[[1]])
[1] 1 2 3
> print(my_list[[2]])
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
```