ITA0448 STATISTICS WITH R PROGRAMMING **P.BARATH**

192121147

Assignment-II

SET-1:

1. Will the following code return any error? State the reason behind your answer and

explain the logic behind the code

```
val &lt;- numeric()

result &lt;- vector(&quot;list&quot;, length(val))

for (index in 1:length(val)) {

result[index] &lt;- val[index] ^ 2

}
```

ANSWER:

The code will not return any error.

The code initializes an empty numeric vector val, and then creates a list of length val called result using the vector function. The for loop iterates over the indices of val, and for each index, it assigns the

squared value of the corresponding element in val to the corresponding element in result.

However, since val is empty, the loop will not execute any iterations, and result will remain a list of length zero. To avoid this, val should be initialized with some values before running the loop.

2. What is the value of equation1(3) for the following R code and explain the logic.

```
&gt; num &lt;- 4

&gt; equation1 &lt;- function (val)

+ {

+ num &lt;- 3

+ num^3 + g (val)

+ }

&gt; equation2 &lt;- function (val)

+ {

+ val*num
+ }
}
```

ANSWER:

The given R code defines two functions equation1 and equation2.

The equation1 function takes an argument val and returns the result of the expression num^3 + g(val), where num is defined as a local variable within the function and

assigned a value of 3, and g(val) is assumed to be a function that takes val as an argument and returns some value. Since g(val) is not defined within the equation1 function, this code would result in an error if called as it is.

On the other hand, the equation2 function takes an argument val and returns the result of the expression val * num, where num is the global variable defined outside both functions and assigned a value of 4.

So, if we call equation1(3), the function first assigns num a local value of 3 and then calculates num^3 + g(3). Since g(val) is undefined, this function would result in an error.

If we call equation2(3), the function returns the value 3 * 4 = 12, as num is the global variable defined outside both functions and assigned a value of 4. Therefore, the value

of equation2(3) is 12.

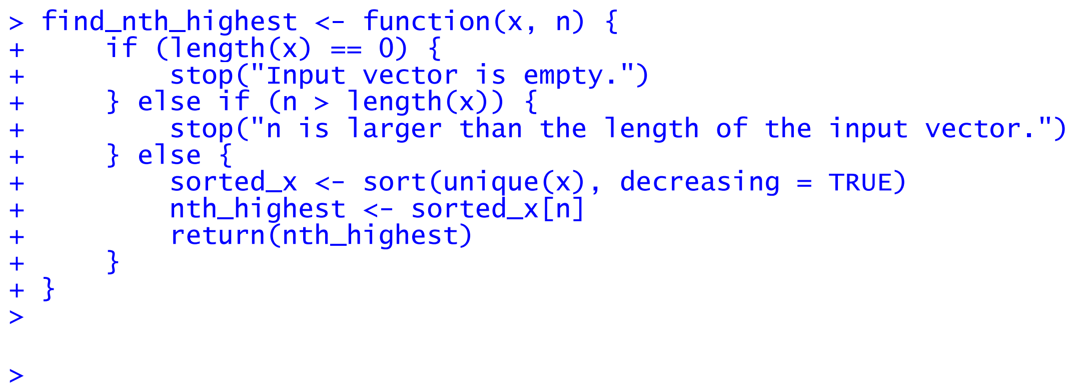
3. Write R function to find nth highest value of a vector in the R program

PROGRAM:

```
        find_nth_highest <- function(x, n) {
 if (length(x) == 0) {
   stop("Input vector is empty.")
 } else if (n > length(x)) {
   stop("n is larger than the length of the input vector.")
 } else {
   sorted_x <- sort(unique(x), decreasing = TRUE)
   nth_highest <- sorted_x[n]
```

```
    return(nth_highest)
  }
}
```

OUTPUT:



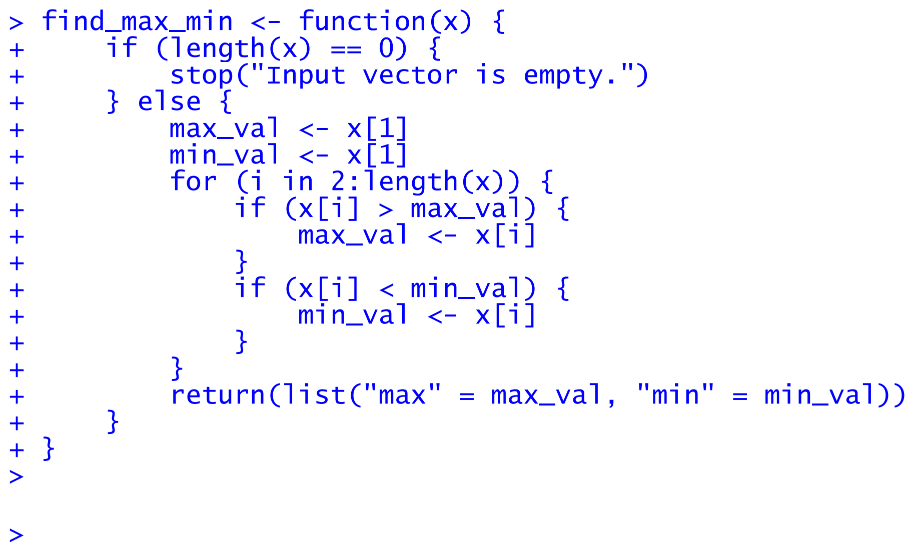
```
> find_nth_highest <- function(x, n) {
+     if (length(x) == 0) {
+         stop("Input vector is empty.")
+     } else if (n > length(x)) {
+         stop("n is larger than the length of the input vector.")
+     } else {
+         sorted_x <- sort(unique(x), decreasing = TRUE)
+         nth_highest <- sorted_x[n]
+         return(nth_highest)
+     }
+ }
>

>
```

5. Write R Program to find maximum and minimum value of a given vector using control statement.

PROGRAM:

```
find_max_min <- function(x) {
  if (length(x) == 0) {
    stop("Input vector is empty.")
  } else {
    max_val <- x[1]
    min_val <- x[1]
    for (i in 2:length(x)) {
      if (x[i] > max_val) {
        max_val <- x[i]
      }
      if (x[i] < min_val) {
        min_val <- x[i]
      }
    }
    return(list("max" = max_val, "min" = min_val))
  }
}
```

OUTPUT:



```
> find_max_min <- function(x) {
+     if (length(x) == 0) {
+         stop("Input vector is empty.")
+     } else {
+         max_val <- x[1]
+         min_val <- x[1]
+         for (i in 2:length(x)) {
+             if (x[i] > max_val) {
+                 max_val <- x[i]
+             }
+             if (x[i] < min_val) {
+                 min_val <- x[i]
+             }
+         }
+         return(list("max" = max_val, "min" = min_val))
+     }
+ }
>

>
```

5. Write R Program to find maximum and minimum value of a given vector using control statement.

PROGRAM:

```
# Define a vector of numbers

my_vector <- c(3, 5, 2, 8, 4, 9, 1)


# Set the initial values of the maximum and minimum to be the first element of the vector

max_value <- my_vector[1]

min_value <- my_vector[1]


# Loop through the vector using a for loop
```

```
for (i in 2:length(my_vector)) {

  # If the current value is greater than the current maximum, update the maximum
  if (my_vector[i] > max_value) {
    max_value <- my_vector[i]
  }

  # If the current value is less than the current minimum, update the minimum
  if (my_vector[i] < min_value) {
    min_value <- my_vector[i]
  }
}

# Print the maximum and minimum values
cat("Maximum value:", max_value, "\n")
cat("Minimum value:", min_value)
```

OUTPUT:

```
> # Define a vector of numbers
> my_vector <- c(3, 5, 2, 8, 4, 9, 1)
> 
> # Set the initial values of the maximum and minimum to be the first eleme
nt of the vector
> max_value <- my_vector[1]
> min_value <- my_vector[1]
> 
> # Loop through the vector using a for loop
> for (i in 2:length(my_vector)) {
+ 
+     # If the current value is greater than the current maximum, update th
e maximum
+     if (my_vector[i] > max_value) {
+         max_value <- my_vector[i]
+     }
+ 
+     # If the current value is less than the current minimum, update the m
inimum
+     if (my_vector[i] < min_value) {
+         min_value <- my_vector[i]
+     }
+ }
> 
```

```
> # Print the maximum and minimum values
> cat("Maximum value:", max_value, "\n")
Maximum value: 9
> cat("Minimum value:", min_value)
Minimum value: 1>

>
```

SET 2 :

1. Create the following matrices (i) Square Matrix (ii) Identity Matrix (iii)diagonal

matrix

PROGRAM:

(i) Square Matrix:

```
# Create a square matrix of size 3x3

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


# Print the matrix

square_matrix
```

OUTPUT:

```
> # Create a square matrix of size 3x3
> square_matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)
>
> # Print the matrix
> square_matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
>

>
```

(ii) Identity Matrix:

```
# Create an identity matrix of size 3x3

identity_matrix <- diag(3)
```

```
# Print the matrix

identity_matrix
```

OUTPUT:

```
> # Create an identity matrix of size 3x3
> identity_matrix <- diag(3)
> 
> # Print the matrix
> identity_matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> 

> 
```

(iii) Diagonal Matrix:

```
# Create a diagonal matrix of size 3x3

diagonal_matrix <- diag(c(1, 2, 3))


# Print the matrix

diagonal_matrix
```

OUTPUT:

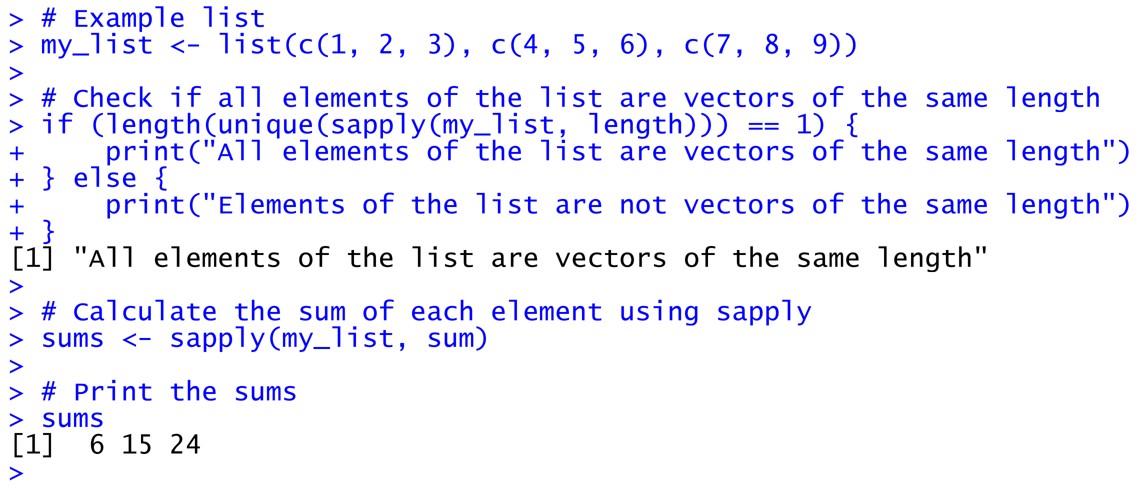
```
> # Create a diagonal matrix of size 3x3
> diagonal_matrix <- diag(c(1, 2, 3))
> 
> # Print the matrix
> diagonal_matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
> 

> 
```

2. Using sapply, check that all elements of the list are vectors of the same length. Also calculate the sum of each element.

PROGRAM:

```
# Example list

my_list <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))


# Check if all elements of the list are vectors of the same length

if (length(unique(sapply(my_list, length))) == 1) {

  print("All elements of the list are vectors of the same length")

} else {

  print("Elements of the list are not vectors of the same length")

}


# Calculate the sum of each element using sapply

sums <- sapply(my_list, sum)


# Print the sums

Sums
```

OUTPUT:



```
> # Example list
> my_list <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))
> 
> # Check if all elements of the list are vectors of the same length
> if (length(unique(sapply(my_list, length))) == 1) {
+     print("All elements of the list are vectors of the same length")
+ } else {
+     print("Elements of the list are not vectors of the same length")
+ }
[1] "All elements of the list are vectors of the same length"
> 
> # Calculate the sum of each element using sapply
> sums <- sapply(my_list, sum)
> 
> # Print the sums
> sums
[1]  6 15 24
> 
```

3. We found out that the blood pressure instrument is under-recording each measure and all measurement incorrect by 0.1. How would you add 0.1 to all values in the blood vector?

PROGRAM:

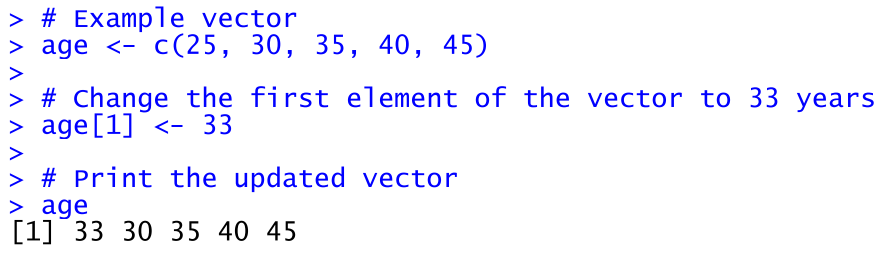
```
# Example vector

blood_pressure <- c(120, 130, 140, 150, 160)


# Add 0.1 to all values in the vector

blood_pressure <- blood_pressure + 0.1


# Print the updated vector

blood_pressure
```

4. We found out that the first patient is 33 years old. How would you change the first element of the vector age to 33 years?

PROGRAM:

```
# Example vector

age <- c(25, 30, 35, 40, 45)


# Change the first element of the vector to 33 years

age[1] <- 33


# Print the updated vector

Age
```

OUTPUT:



```
> # Example vector
> age <- c(25, 30, 35, 40, 45)
>
> # Change the first element of the vector to 33 years
> age[1] <- 33
>
> # Print the updated vector
> age
[1] 33 30 35 40 45
```

```
>

>
```

5. Suppose A = [1 1 3 5 2 6 −2 −1 −3] (a) Check that A 3 = 0 where 0 is a 3 × 3 matrix with every entry equal to 0. (b) Replace the third column of A by the sum of the second and third columns

PROGRAM:

A)

# Define the matrix A

A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)

# Create a 3x3 submatrix from the first nine elements of A

A_sub <- matrix(A[1:9], nrow = 3)

# Check if A_sub is a zero matrix

all(A_sub == 0)

OUTPUT:

```
> # Define the matrix A
> A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)
> 
> # Create a 3x3 submatrix from the first nine elements of A
> A_sub <- matrix(A[1:9], nrow = 3)
> 
> # Check if A_sub is a zero matrix
> all(A_sub == 0)
[1] FALSE
>

>
```

B)

# Define the matrix A

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


# Replace the third column of A by the sum of the second and third columns

A[,3] <- A[,2] + A[,3]


# Print the updated matrix A

A
```

OUTPUT:

```
> # Define the matrix A
> A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)
> 
> # Replace the third column of A by the sum of the second and third colum
ns
> A[,3] <- A[,2] + A[,3]
```

1.a. The numbers below are the first ten days of rainfall amounts in 1996. Read them into a vector using

the c() function

1.1 0.6 33.8 1.9 9.6 4.3 33.7 0.3 0.0 0.1

```
rainfall <- c(1.1, 0.6, 33.8, 1.9, 9.6, 4.3, 33.7, 0.3, 0.0, 0.1)
```

b) To find the mean rainfall and standard deviation, we can use the `mean()` and `sd()` functions in R, respectively:

PROGRAM:

```
mean(rainfall)   # Mean rainfall
```

```
sd(rainfall)     # Standard deviation of rainfall
```

OUTPUT:

```
[1] 7.54

[1] 13.20124
```

c) To find the day with the highest rainfall, we can use the `which.max()` function, which returns the index of the maximum value in a vector:

PROGRAM:

```
which.max(rainfall)  # Index of the day with the highest rainfall
```

OUTPUT:

```
[1] 3
```

d) To find the 18th letter of the alphabet, we can use the `letters` vector in R:

PROGRAM:

```
letters[18]
```

OUTPUT:

```
[1] "r"
```

e) To find the last letter of the alphabet, we can use the **LETTERS** vector in R:

PROGRAM:

```
LETTERS[26]
```

OUTPUT:

```
[1] "Z"
```