**UNIT 2: Looping, Control Structures and Recursion in R**

**1. Factorial**

factorial\_calc <- function(n) {

if (!is.numeric(n) || n < 0 || floor(n) != n) {

return("Error: Please enter a non-negative integer.")

} else if (n == 0) {

return(1)

} else {

fact <- 1

for (i in 1:n) {

fact <- fact \* i

}

return(fact)

}

}

num <- as.integer(readline("Enter a non-negative integer: "))

result <- factorial\_calc(num)

cat("Factorial:", result, "\n")

**Output:**

****

**2. Fibonacci Series**

fibonacci\_seq <- function(limit) {

if (limit < 1) return("Error: Enter a positive number.")

fib <- c(0, 1) # Initialize sequence

while (TRUE) {

next\_num <- tail(fib, 1) + tail(fib, 2)[1]

if (next\_num > limit) break

fib <- c(fib, next\_num)

}

cat("Fibonacci Sequence:", fib, "\n")

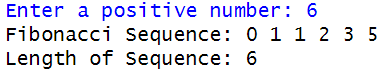
cat("Length of Sequence:", length(fib), "\n")

}

num <- suppressWarnings(as.integer(readline("Enter a positive number: ")))

fibonacci\_seq(num)

**Output:**

****

**3. Student Grade**

assign\_grade <- function(score) {

if (score < 0 || score > 100) {

return("Error: Enter a valid score between 0 and 100.")

} else if (score >= 90) {

grade <- "A"

} else if (score >= 80) {

grade <- "B"

} else if (score >= 70) {

grade <- "C"

} else if (score >= 60) {

grade <- "D"

} else {

grade <- "F"

}

return(paste("Grade:", grade))

}

score <- suppressWarnings(as.integer(readline("Enter your score (0-100): ")))

cat(assign\_grade(score), "\n")

**Output:**

****

**4. Mean of Vectors**

vectors <- list(c(1, 2, 3, "a"), c(10, 20, 30, NA), c(5, 15, 25, "b", 35), c(100, 200, 300))

calculate\_mean <- function(vec) {

numeric\_vec <- as.numeric(vec)

mean\_val <- mean(numeric\_vec, na.rm = TRUE)

return(mean\_val)

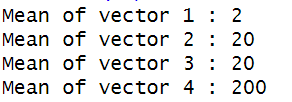
}

for (i in seq\_along(vectors)) {

cat("Mean of vector", i, ":", calculate\_mean(vectors[[i]]), "\n")

}

**Output:**

****

**5. Name Age Score Data frame**

df <- data.frame(Name = c("Alice", "Bob", "Charlie", "David"),

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

Score = c(85, 90, 75, 88))

for (i in 1:nrow(df)) {

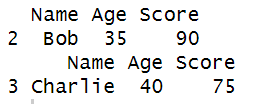
if (df$Age[i] > 30) {

print(df[i, ])

}

}

**Output:**

****

**6. Arithmetic Operations**

a <- as.numeric(readline("Enter first number: "))

b <- as.numeric(readline("Enter second number: "))

if (is.na(a) || is.na(b)) {

cat("Error: Enter valid numbers.\n")

} else {

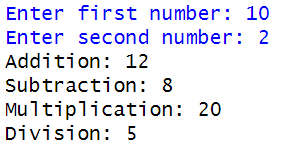
cat("Addition:", a + b, "\nSubtraction:", a - b,

"\nMultiplication:", a \* b,

"\nDivision:", ifelse(b != 0, a / b, "Error (division by zero)"), "\n")

}

**Output:**

****

**7. Attendance Requirement**

students <- data.frame(

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

Attendance = c(TRUE, FALSE, TRUE, FALSE), # TRUE if attendance is met

Exam\_Passed = c(FALSE, TRUE, TRUE, FALSE) # TRUE if exam is passed

)

for (i in 1:nrow(students)) {

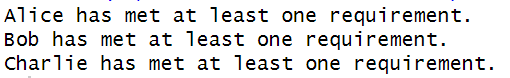
if (students$Attendance[i] || students$Exam\_Passed[i]) {

cat(students$Name[i], "has met at least one requirement.\n")

}

}

**Output:**

****

**8. Mean, Median, Mode**

calculate\_stats <- function(vec) {

mean\_val <- mean(vec, na.rm = TRUE) # Calculate mean

median\_val <- median(vec, na.rm = TRUE) # Calculate median

unique\_vals <- unique(vec)

mode\_val <- unique\_vals[which.max(tabulate(match(vec, unique\_vals)))]

return(list(Mean = mean\_val, Median = median\_val, Mode = mode\_val))

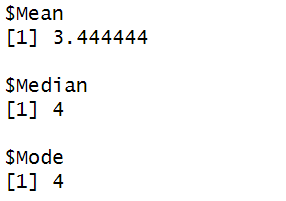
}

num\_vec <- c(1, 2, 2, 3, 4, 4, 4, 5, 6)

stats <- calculate\_stats(num\_vec)

print(stats)

**Output:**

****

**9. Factorial Using lapply()**

factorial\_recursive <- function(n) {

if (n == 0) return(1)

return(n \* factorial\_recursive(n - 1))

}

num\_vector <- c(3, 5, 7, 0)

factorials <- sapply(num\_vector, factorial\_recursive)

print(factorials)

**Output:**

****

**10. Recursive Fibonacci**

fibonacci\_recursive <- function(n) {

if (n <= 0) return("Error: Enter a positive integer.")

if (n == 1) return(0)

if (n == 2) return(1)

return(fibonacci\_recursive(n - 1) + fibonacci\_recursive(n - 2))

}

n <- as.integer(readline("Enter a positive integer (n): "))

if (is.na(n) || n <= 0) {

cat("Error: Please enter a valid positive integer.\n")

} else {

cat("Fibonacci number at position", n, "is:", fibonacci\_recursive(n), "\n")

}

**Output:**

****

**11. Odd numbers get +2 & Even numbers get \*2**

math\_operation <- function(x, multiplier = 2) {

if (x %% 2 == 0) {

result <- x \* multiplier

} else {

result <- x + multiplier

}

return(list(Number = x, Result = result))

}

num\_vector <- c(3, 6, 9)

results <- list()

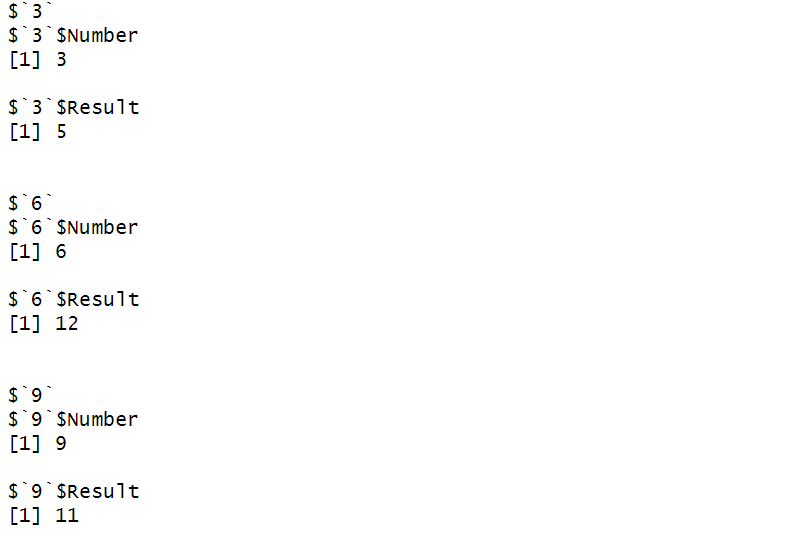
for (i in num\_vector) {

results[[as.character(i)]] <- math\_operation(i)

}

print(results)

**Output:**

****

**12. Area of a Rectangle**

rectangle\_area <- function(length = 5, width = 3) {

return(length \* width) # Area formula: length × width

}

default\_area <- rectangle\_area()

cat("Area with default values:", default\_area, "\n")

custom\_area <- rectangle\_area(8, 4)

cat("Area with custom values (8x4):", custom\_area, "\n")

**Output:**

****

**13. Prime Number or not**

is\_prime <- function(n) {

if (n < 2) {

return(FALSE)

}

for (i in 2:sqrt(n)) {

if (n %% i == 0) {

return(FALSE)

}

}

return(TRUE)

}

num <- as.integer(readline("Enter an integer: "))

if (is.na(num) || num < 1) {

cat("Error: Please enter a valid positive integer.\n")

} else if (is\_prime(num)) {

cat(num, "is a prime number.\n")

} else {

cat(num, "is not a prime number.\n")

}

**Output:**

****

**14. Sum a Vector in R**

sum\_recursive <- function(vec) {

if (length(vec) == 0) return(0)

return(vec[1] + sum\_recursive(vec[-1]))

}

sample\_vec <- c(3, 5, 7, 2, 8)

result <- sum\_recursive(sample\_vec)

cat("Sum of the vector:", result, "\n")

**Output:**

****

**15. Score and Grade**

assign\_grade <- function(score) {

if (score >= 90 && score <= 100) {

return("A")

} else if (score >= 80) {

return("B")

} else if (score >= 70) {

return("C")

} else if (score >= 60) {

return("D")

} else {

return("F")

}

}

score <- as.integer(readline("Enter the score: "))

if (is.na(score) || score < 0 || score > 100) {

cat("Error: Please enter a valid score between 0 and 100.\n")

} else {

cat("Grade:", assign\_grade(score), "\n")

}

**Output:**

****

**16. Classifying Numbers as Positive, Negative & Zero**

classify\_numbers <- function(vec) {

result <- sapply(vec, function(x) {

if (x > 0) {

return("positive")

} else if (x < 0) {

return("negative")

} else {

return("zero")

}

})

return(result)

}

num\_vector <- c(-3, 5, 0, -7, 2, 0, 8)

classified\_vec <- classify\_numbers(num\_vector)

print(classified\_vec)

**Output:**

****

**17. Categorizing Fruits, Vegetables & Electronics**

items\_list <- list(

Fruits = c("Apple", "Banana", "Orange"),

Vegetables = c("Carrot", "Spinach", "Potato", "Tomato"),

Electronics = c("Laptop", "Phone")

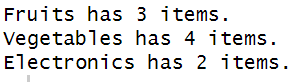
)

for (category in names(items\_list)) {

cat(category, "has", length(items\_list[[category]]), "items.\n")

}

**Output:**

****

**18. Duplicated Product Combinations & Unique Customer-Product Pairs**

customers <- c("Alice", "Bob", "Alice", "Charlie", "Bob", "Alice", "Charlie")

products <- c("Laptop", "Phone", "Laptop", "Tablet", "Phone", "Tablet", "Tablet")

df <- data.frame(Customer = customers, Product = products)

duplicates <- df[duplicated(df), ] # Find duplicates

unique\_pairs <- unique(df) # Get unique customer-product pairs

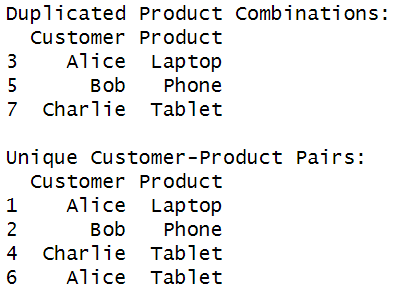
cat("Duplicated Product Combinations:\n")

print(duplicates)

cat("\nUnique Customer-Product Pairs:\n")

print(unique\_pairs)

**Output:**

****

**19. Duplicates and Unique Treatments**

patients <- c("John", "Emma", "John", "Michael", "Emma", "John", "Michael")

treatments <- c("Aspirin", "Ibuprofen", "Aspirin", "Paracetamol", "Ibuprofen", "Paracetamol", "Paracetamol")

df <- data.frame(Patient = patients, Treatment = treatments)

duplicates <- df[duplicated(df), ] # Find duplicate rows

unique\_treatments <- unique(df) # Get unique patient-treatment combinations

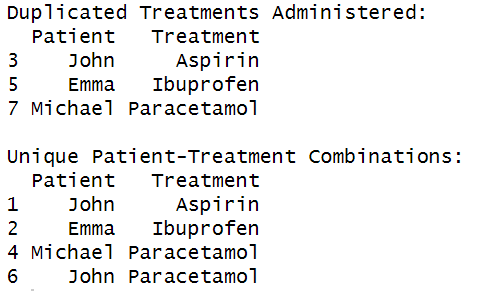
cat("Duplicated Treatments Administered:\n")

print(duplicates)

cat("\nUnique Patient-Treatment Combinations:\n")

print(unique\_treatments)

**Output:**

****

**20. Patient Data Frame**

patient\_IDs <- c(101, 102, 101, 103, 102, 101, 103)

treatments <- c("Aspirin", "Ibuprofen", "Aspirin", "Paracetamol", "Ibuprofen", "Paracetamol", "Paracetamol")

df <- data.frame(Patient\_ID = patient\_IDs, Treatment = treatments)

duplicates <- df[duplicated(df), ] # Extract duplicate records

unique\_treatments <- unique(df) # Get unique patient-treatment combinations

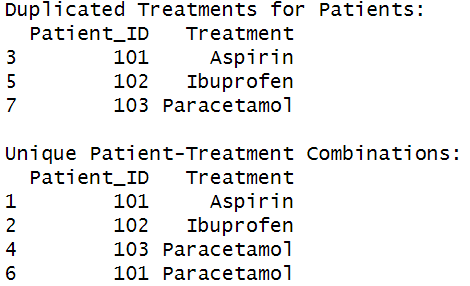
cat("Duplicated Treatments for Patients:\n")

print(duplicates)

cat("\nUnique Patient-Treatment Combinations:\n")

print(unique\_treatments)

**Output:**

****