Looping, Control Structures, Functions, Recursion in R

1. Factorial Calculation using For Loop

Script:

```
number <- as.integer(readline(prompt = "Enter a number: "))
if (number < 0) {
  cat("Error: Factorial of a negative number doesn't exist.\n")
} else {
  factorial <- 1
  for (i in 1:number) {
    factorial <- factorial * i
  }
  cat("Factorial of", number, "is", factorial, "\n")
}
```

Output:

```
Enter a number: 5
Factorial of 5 is 120
```

2. Fibonacci Sequence using While Loop

Script:

```
limit <- as.integer(readline(prompt = "Enter the limit: "))
fib_seq <- c(0, 1)
i <- 2
while (fib_seq[i] + fib_seq[i-1] <= limit) {
  fib_seq <- c(fib_seq, fib_seq[i] + fib_seq[i-1])
  i <- i + 1
}
cat("Fibonacci Sequence:", fib_seq, "\n")
cat("Length of sequence:", length(fib_seq), "\n")
```

Output:

Enter the limit: 20
Fibonacci Sequence: 0 1 1 2 3 5 8 13
Length of sequence: 8

3. Grade Assignment Based on Score

Script:

```
score <- as.integer(readline(prompt = "Enter the score: "))
if (score >= 90 & score <= 100) {
  grade <- "A"
} else if (score >= 80 & score < 90) {
  grade <- "B"
} else if (score >= 70 & score < 80) {
  grade <- "C"
} else if (score >= 60 & score < 70) {
  grade <- "D"
} else if (score >= 0 & score < 60) {
  grade <- "F"
} else {
  grade <- "Invalid score"
}
cat("Grade:", grade, "\n")
```

Output:

Enter the score: 85
Grade: B

4. Mean Calculation Ignoring Non-Numeric Values

Script:

```
list_of_vectors <- list(c(1, 2, 3, "a"), c(4, 5, 6), c(7, "b", 9))
for (vec in list_of_vectors) {
  numeric_values <- as.numeric(vec)
  mean_value <- mean(numeric_values, na.rm = TRUE)
  cat("Mean of vector:", mean_value, "\n")
```

```
}
```

Output:

```
Mean of vector: 2
Mean of vector: 5
Mean of vector: 8
```

5. Print Rows Where Age > 30

Script:

```
df <- data.frame(Name = c("Alice", "Bob", "Charlie"), Age = c(25, 35, 30))
for (i in 1:nrow(df)) {
  if (df$Age[i] > 30) {
    print(df[i, ])
  }
}
```

Output:

```
  Name Age
2  Bob  35
```

6. Basic Arithmetic Operations

Script:

```
num1 <- as.numeric(readline(prompt = "Enter first number: "))
num2 <- as.numeric(readline(prompt = "Enter second number: "))
cat("Addition:", num1 + num2, "\n")
cat("Subtraction:", num1 - num2, "\n")
cat("Multiplication:", num1 * num2, "\n")
cat("Division:", num1 / num2, "\n")
```

Output:

```
Enter first number: 10
Enter second number: 5
```

Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2

7. Attendance or Exam Pass Check

Script:

```
attendance <- as.logical(readline(prompt = "Enter attendance status (TRUE/FALSE): "))
exam_passed <- as.logical(readline(prompt = "Enter exam status (TRUE/FALSE): "))
if (attendance || exam_passed) {
  cat("Student has met at least one requirement.\n")
} else {
  cat("Student has not met the requirements.\n")
}
```

Output:

```
Enter attendance status (TRUE/FALSE): TRUE
Enter exam status (TRUE/FALSE): FALSE
Student has met at least one requirement.
```

8. Mean, Median, and Mode Function

Script:

```
calculate_stats <- function(vec) {
  mode_val <- as.numeric(names(sort(table(vec), decreasing = TRUE)[1]))
  list(mean = mean(vec), median = median(vec), mode = mode_val)
}
result <- calculate_stats(c(1, 2, 2, 3, 4))
print(result)
```

Output:

```
$mean
[1] 2.4
```

```
$median
[1] 2

$mode
[1] 2
```

9. Recursive Factorial Calculation

Script:

```
factorial_recursive <- function(n) {
  if (n == 0) return(1)
  else return(n * factorial_recursive(n - 1))
}
nums <- c(3, 4, 5)
factorials <- sapply(nums, factorial_recursive)
print(factorials)
```

Output:

```
3  4  5
6 24 120
```

10. Recursive Fibonacci Number

Script:

```
fibonacci_recursive <- function(n) {
  if (n <= 1) return(n)
  return(fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2))
}
n <- as.integer(readline(prompt = "Enter n: "))
cat("Fibonacci number:", fibonacci_recursive(n), "\n")
```

Output:

```
Enter n: 7
Fibonacci number: 13
```

11. For Loop with Condition and Function

Script:

```
add_numbers <- function(x, y = 2) return(x + y)
vec <- c(1, 2, 3, 4, 5)
for (num in vec) {
  if (num %% 2 == 0) {
    result <- add_numbers(num)
    print(list(original = num, result = result))
  }
}
```

Output:

```
$original
[1] 2

$result
[1] 4

$original
[1] 4

$result
[1] 6
```

12. Area of a Rectangle Function

Script:

```
area_rectangle <- function(length = 5, width = 3) return(length * width)
cat("Default area:", area_rectangle(), "\n")
cat("Custom area:", area_rectangle(10, 4), "\n")
```

Output:

```
Default area: 15
Custom area: 40
```

13. Prime Number Check

Script:

```
num <- as.integer(readline(prompt = "Enter a number: "))
is_prime <- function(n) {
 if (n <= 1) return(FALSE)
 for (i in 2:sqrt(n)) {
  if (n %% i == 0) return(FALSE)
 }
 return(TRUE)
}
if (is_prime(num)) cat(num, "is a prime number.\n")
else cat(num, "is not a prime number.\n")
```

Output:

```
Enter a number: 7
7 is a prime number.
```

14. Recursive Sum of Vector Elements

Script:

```
sum_recursive <- function(vec) {
 if (length(vec) == 0) return(0)
 return(vec[1] + sum_recursive(vec[-1]))
}
vec <- c(1, 2, 3, 4, 5)
cat("Sum of vector:", sum_recursive(vec), "\n")
```

Output:

```
Sum of vector: 15
```

15. Grade Assignment Based on Score

Script:

```
score <- as.integer(readline(prompt = "Enter score: "))
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"
cat("Grade:", grade, "\n")
```

Output:

```
Enter score: 75
Grade: C
```

16. Replace Numbers with Positive, Negative, or Zero

Script:

```
replace_sign <- function(vec) {
  sapply(vec, function(x) {
    if (x > 0) return("positive")
    else if (x < 0) return("negative")
    else return("zero")
  })
}
result <- replace_sign(c(-1, 0, 2, 3, -4))
print(result)
```

Output:

```
[1] "negative" "zero" "positive" "positive" "negative"
```

17. Loop Over Categories and Count Items

Script:

```
categories <- list(fruits = c("apple", "banana"), vegetables = c("carrot"), electronics =
c("phone", "laptop", "tablet"))
for (category in names(categories)) {
```

```
  cat("Category:", category, "- Number of items:", length(categories[[category]]), "\n")
}
```

Output:

```
Category: fruits - Number of items: 2
Category: vegetables - Number of items: 1
Category: electronics - Number of items: 3
```

18. Data Frame with Duplicated Products

Script:

```
customers <- c("John", "Jane", "John", "Doe")
products <- c("Laptop", "Phone", "Laptop", "Tablet")
df <- data.frame(Customer = customers, Product = products)
duplicates <- df[duplicated(df), ]
unique_pairs <- unique(df)
print(duplicates)
print(unique_pairs)
```

Output:

```
  Customer Product
3     John  Laptop
  Customer Product
1     John  Laptop
2     Jane   Phone
4      Doe  Tablet
```

19. Data Frame with Duplicated Treatments

Script:

```
patients <- c("Alice", "Bob", "Alice", "Eve")
treatments <- c("X-Ray", "MRI", "X-Ray", "CT Scan")
df <- data.frame(Patient = patients, Treatment = treatments)
duplicates <- df[duplicated(df), ]
unique_pairs <- unique(df)
```

```
print(duplicates)
print(unique_pairs)
```

Output:

```
  Patient Treatment
3   Alice     X-Ray
  Patient Treatment
1   Alice     X-Ray
2     Bob      MRI
4     Eve   CT Scan
```

20. Data Frame for Patient-Treatment Analysis

Script:

```
patient_ids <- c(1, 2, 1, 3)
treatments <- c("A", "B", "A", "C")
df <- data.frame(PatientID = patient_ids, Treatment = treatments)
duplicates <- df[duplicated(df), ]
unique_combinations <- unique(df)
print(duplicates)
print(unique_combinations)
```

Output:

```
  PatientID Treatment
3         1         A
  PatientID Treatment
1         1         A
2         2         B
4         3         C
```