Exercise 1: Basic Arithmetic Operations

You are working as a cashier at a grocery store. Your task is to create a program that simulates the checkout process for a customer's shopping cart. The program should calculate the total cost of the items, including tax, and provide a detailed receipt.

i. Define a list of products, each represented as a dictionary with keys: "name", "price", and "quantity".

ii. Allow the cashier to input the products in the customer's shopping cart, including the name, price, and quantity of each item.

iii. Calculate the subtotal (price * quantity) for each item and display a detailed receipt with product names, quantities, prices, and subtotals.

iv. Calculate the total cost of the items in the cart before tax.

v. Apply a tax rate (e.g., 8%) to the total cost to calculate the tax amount.

vi. Calculate the final total cost by adding the tax amount to the total cost before tax.

```
products <- list(
list(name = "Apple", price = 0.5),
list(name = "Banana", price = 0.3),
list(name = "Milk", price = 2),
list(name = "Bread", price = 1.5),
list(name = "Eggs", price = 2.5)
)

shopping_cart <- list()

cart_items_to_add <- list(
list(name = "Apple", quantity = 3),
list(name = "Milk", quantity = 2)
)

for (item in cart_items_to_add)
{
product_name <- item$name
quantity <- item$quantity

product <- NULL
for (p in products)
{
if (p$name == product_name)
{
product <- p
break
}
}
if (!is.null(product))
{
cart_item <- list(name = product$name, price = product$price, quantity = quantity)
shopping_cart <- c(shopping_cart, list(cart_item))
cat("Item added to cart.\n")
} else {
cat("Product not found.\n")
}
}
```

```
subtotal <- 0
cat("\nReceipt:\n")
for (item in shopping_cart)
{
item_subtotal <- item$price * item$quantity
cat(sprintf("%s (%d units) - Price: $%.2f - Subtotal: $%.2f\n", item$name, item$quantity,
item$price, item_subtotal))
subtotal <- subtotal + item_subtotal
}
tax_rate <- 0.08
tax_amount <- subtotal * tax_rate
total_cost_before_tax <- subtotal
total_cost <- total_cost_before_tax + tax_amount

cat("\nSubtotal: $%.2f\n", subtotal)
cat("Tax Amount (8%): $%.2f\n", tax_amount)
cat("Total Cost: $%.2f\n", total_cost)
```

Exercise 2: Loops Operations

You have been tasked with creating a program that calculates and assigns grades for students enrolled in multiple courses. The program will take input for the marks obtained by 10 students in 5 different courses, compute the total and average marks for each student, and assign corresponding grades based on their average performance.
Declare constants for the number of students (num_students) and the number of courses (num_courses).
Initialize an empty list to store student information.
For each student:
• Input the student's name.
• Input marks for each of the 5 courses.
• Calculate the total marks and average marks.
• Determine the grade based on the average marks using a grading scale.

• Display the student information, including their name, individual course marks, total marks,average marks, and the assigned grade.

```
num_students <- 5
num_courses <- 5

student_names <- c("Arun Rahul", "Bheem Kumar","Raj jumar","Jahal A R","Suresh")
course_marks <- matrix
(c(85, 92, 78, 88, 95,
75, 80, 85, 70, 60,
100,78,56,34,56,
78,45,67,89,90,
89,80,67,78,90
), nrow = num_students, byrow = TRUE)

student_records <- list()

for (student_index in 1:num_students) {
student_name <- student_names[student_index]
total_marks <- sum(course_marks[student_index, ])
average_marks <- total_marks / num_courses
```

```
grade <- ifelse(average_marks >= 90, "A",
ifelse(average_marks >= 80, "B",
ifelse(average_marks >= 70, "C",
ifelse(average_marks >= 60, "D", "F"))))

student_record <- list(name = student_name, marks = course_marks[student_index, ],
total = total_marks, average = average_marks, grade = grade)
student_records <- c(student_records, list(student_record))
}

cat("\nStudent Grade Report:\n")
for (student_record in student_records)
{
cat("\nName:", student_record$name, "\n")
cat("Marks:", student_record$marks, "\n")
cat("Total Marks:", student_record$total, "\n")
cat("Average Marks:", student_record$average, "\n")
cat("Grade:", student_record$grade, "\n")
}
```

Output of the program:

```
Student Grade Report:
Name: Arun Rahul
Marks: 85 92 78 88 95
Total Marks: 438
Average Marks: 87.6
Grade: B
Name: Bheem Kumar
Marks: 75 80 85 70 60
Total Marks: 370
Average Marks: 74
Grade: C
Name: Raj jumar
Marks: 100 78 56 34 56
Total Marks: 324
Average Marks: 64.8
Grade: D
```

Exercise 3: Conditional statement, Loops and Functions

You are developing a library management system that needs a fine calculation feature. Write a program that takes the number of days a book is overdue and calculates the fine amount based on the library's policy. The policy states that for the first 7 days, there is no fine. After 7 days, a fixed fine per day is charged. Additionally, there's a cap on the fine amount after 30 days.
Input the number of days the book is overdue.
• Use conditional statements to calculate the fine amount based on the library's policy.

• Display the fine amount along with a message indicating whether the fine is within the cap or exceeded it.

```
calculate_fine <- function(days_overdue)
{
if (days_overdue <= 7)
{
fine <- 0
} else if (days_overdue <= 30)
{
fine_per_day <- 2
fine <- (days_overdue - 7) * fine_per_day
} else {
fine_cap <- 50
fine <- fine_cap
}
return(fine)
}

days_overdue <- as.integer(readline("Enter the number of days the book is overdue: "))
fine_amount <- calculate_fine(days_overdue)
cat("Fine Amount:", fine_amount, "\n")

if (fine_amount == 0)
{
cat("No fine.\n")
} else {
if (days_overdue > 30)
{
cat("Fine exceeds the maximum cap\n")
} else {
cat("Please pay the fine\n")
}
}
```

Output:
Enter the number of days the book is overdue: 20
Fine Amount: 26
Please pay the fine within the specified period

Exercise 4: arrays and Functions:

You are developing an inventory management system for a small store. The system needs to handle inventory items and their quantities. Write a program that uses arrays to store inventory items and their quantities, and includes functions to add new items, update quantities, and display the inventory.

- Define an array to store inventory items.
- Define an array to store corresponding quantities.
- Implement functions to:
  - Add a new item along with its quantity.
  - Update the quantity of an existing item.
  - Display the inventory items and quantities.
  - Use the functions to manage the inventory and handle user interactions.

```
inventory_items <- character(0)
inventory_quantities <- numeric(0)

add_item <- function(item, quantity) {
inventory_items <<- c(inventory_items, item)
inventory_quantities <<- c(inventory_quantities, quantity)
cat("Item added to inventory.\n")
}

update_quantity <- function(item, new_quantity) {
if (item %in% inventory_items) {
item_index <- which(inventory_items == item)
inventory_quantities[item_index] <<- new_quantity
cat("Quantity updated.\n")
} else {
cat("Item not found in inventory.\n")
}
}

display_inventory <- function() {
cat("Inventory Items and Quantities:\n")
for (i in 1:length(inventory_items)) {
cat(sprintf("%s: %d\n", inventory_items[i], inventory_quantities[i]))
}
}

while (TRUE) {
cat("\n1. Add Item\n2. Update Quantity\n3. Display Inventory\n4. Exit\n")
choice <- as.integer(readline("Enter your choice: "))
if (choice == 1) {
item <- readline("Enter item name: ")
quantity <- as.integer(readline("Enter quantity: "))
add_item(item, quantity)
} else if (choice == 2) {
item <- readline("Enter item name: ")
new_quantity <- as.integer(readline("Enter new quantity: "))
update_quantity(item, new_quantity)
} else if (choice == 3) {
display_inventory()
```

```
} else if (choice == 4) {
cat("Exiting the program. Goodbye!\n")
break
} else {
cat("Invalid choice. Please try again.\n")
}
}
```

Lab5: Dataframe

You are working as an educational analyst and need to analyze the performance of students in a school. You have data on student names, their scores in different subjects, and attendance. Write a program that uses data frames to manage and analyze student data, including calculating average scores, identifying students with low attendance, and generating a report.
Create a data frame to store student information with columns: "Name", "Math_Score", "Science_Score", "History_Score", "Attendance".
Implement functions to:
• Calculate the average scores for each student.
• Identify students with attendance below a certain threshold.
• Generate a report with student names, average scores, and attendance status.

• Use the functions to analyse student performance and generate the report.

```
#Load dplyr: tools>install packages>type dplyr & install

library(dplyr)

student_data <- data.frame(
Name = character(0),
Math_Score = numeric(0),
Science_Score = numeric(0),
History_Score = numeric(0),
Attendance = numeric(0)
)

add_student <- function(name, math_score, science_score, history_score, attendance) {
new_student <- data.frame(
Name = name,
Math_Score = math_score,
Science_Score = science_score,
History_Score = history_score,
Attendance = attendance
)
student_data <<- bind_rows(student_data, new_student)
cat("Student information added.\n")
}

calculate_average_scores <- function() {
avg_scores <- student_data %>%
mutate(Average_Score = (Math_Score + Science_Score + History_Score) / 3) %>%
select(Name, Average_Score)
return(avg_scores)
}

identify_low_attendance <- function(threshold) {
```

```
low_attendance <- student_data %>%
filter(Attendance < threshold) %>%
select(Name, Attendance)
return(low_attendance)
}

generate_report <- function() {
avg_scores <- calculate_average_scores()
low_attendance <- identify_low_attendance(70)
report <- merge(avg_scores, low_attendance, by = "Name", all = TRUE)
report$Attendance[is.na(report$Attendance)] <- 100
cat("Performance Report:\n")
print(report)
}

while (TRUE) {
cat("\n1. Add Student\n2. Generate Report\n3. Exit\n")
choice <- as.integer(readline("Enter your choice: "))
if (choice == 1) {
name <- readline("Enter student name: ")
math_score <- as.numeric(readline("Enter math score: "))
science_score <- as.numeric(readline("Enter science score: "))
history_score <- as.numeric(readline("Enter history score: "))
attendance <- as.numeric(readline("Enter attendance percentage: "))
add_student(name, math_score, science_score, history_score, attendance)
} else if (choice == 2) {
generate_report()
} else if (choice == 3) {
cat("Exiting the program. Goodbye!\n")
break
} else {
cat("Invalid choice. Please try again.\n")
}
}
```

Lab 6 :

You are a data analyst at a retail company that sells products online. The company is interested in predicting sales for the upcoming months to better manage inventory and plan marketing strategies. As part of your role, you need to develop a program that utilizes time series analysis to forecast sales based on a historical sales dataset.

Write an R program to forecast sales for the next three months using time series analysis techniques. The program should perform the following steps:

- Load the required libraries, including the forecast package.
- Create a data frame with two columns: Month and Sales. The Month column should contain a sequence of dates from January 2023 to June 2023 (inclusive), and the Sales column should contain the corresponding sales amounts (12000, 15000, 18000, 16000, 20000, 22000).
- Convert the sales data into a time series object with a monthly frequency.
- Fit an ARIMA (AutoRegressive Integrated Moving Average) model to the sales time series using the auto.arima() function.
- Forecast sales for the next three months using the fitted ARIMA model and the forecast() function.
- Display the forecasted sales results, including point forecasts and prediction intervals.

```
#install forecast package in tools
library(forecast)

sales_data <- data.frame(
Month = seq(as.Date("2023-01-01"), as.Date("2023-06-01"), by = "months"),
Sales = c(12000, 15000, 18000, 16000, 20000, 22000)
)

sales_ts <- ts(sales_data$Sales, frequency = 12)

arima_model <- auto.arima(sales_ts)

forecast_result <- forecast(arima_model, h = 3)

print(forecast_result)
```

Lab 7 : Customer Purchase Analysis for E-commerce Company (Enhanced)

You are a data analyst working for an e-commerce company that specializes in selling a variety of products online. The company aims to analyze customer purchase data comprehensively to gain insights into customer behavior and spending patterns.
Your goal is to develop a R program that performs an in-depth analysis of customer purchase data. You will calculate various statistical measures and generate visualizations to understand the distribution of purchase amounts among customers.
Note: Load the necessary libraries, including the dplyr and ggplot2 packages.
Given the example customer purchase data provided below, create a data frame named purchase_data with two columns: CustomerID and PurchaseAmount.
Calculate and display the following statistical measures:

- Mean (average) purchase amount
- Median purchase amount
- Standard deviation of purchase amounts
- 1st quartile (25th percentile) of purchase amounts
- 3rd quartile (75th percentile) of purchase amounts

```
library(dplyr)
library(ggplot2)

purchase_data <- data.frame(
  CustomerID = c(101, 102, 103, 104, 105),
  PurchaseAmount = c(150, 200, 120, 300, 80)
)

mean_purchase <- mean(purchase_data$PurchaseAmount)
median_purchase <- median(purchase_data$PurchaseAmount)
sd_purchase <- sd(purchase_data$PurchaseAmount)
q1_purchase <- quantile(purchase_data$PurchaseAmount, probs = 0.25)
q3_purchase <- quantile(purchase_data$PurchaseAmount, probs = 0.75)

cat("Mean Purchase Amount:", mean_purchase, "\n")
cat("Median Purchase Amount:", median_purchase, "\n")
cat("Standard Deviation of Purchase Amounts:", sd_purchase, "\n")
cat("1st Quartile of Purchase Amounts:", q1_purchase, "\n")
cat("3rd Quartile of Purchase Amounts:", q3_purchase, "\n")
```

Lab 8: Matrix Manipulation in R

Write an R program that generates two matrices, matrix_A and matrix_B, and conducts operations including element-wise addition, scalar multiplication, matrix transpose, and multiplication.

```
matrix_A <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)
matrix_B <- matrix(c(9, 8, 7, 6, 5, 4, 3, 2, 1), nrow = 3, ncol = 3, byrow = TRUE)

sum_matrix <- matrix_A + matrix_B
scaled_matrix <- matrix_A * 2

transposed_A <- t(matrix_A)
product_matrix <- matrix_A %*% matrix_B

sum_matrix_A <- sum(matrix_A)
mean_matrix_B <- mean(matrix_B)
sd_matrix_B <- sd(matrix_B)

cat("Matrix A:\n")
print(matrix_A)

cat("\nMatrix B:\n")
print(matrix_B)

cat("\n element wise Addition:\n")
print(sum_matrix)

cat("\n Scalar Multiplication:\n")
print(scaled_matrix)

cat("\nTranspose ofA:\n")
print(transposed_A)

cat("\nMatrix Multiplication:\n")
print(product_matrix)

library(ggplot2)
library(reshape2)
row_sums <- rowSums(matrix_B)
row_names <- paste("Row", 1:3)
barplot_data <- data.frame(Row = row_names, Sum = row_sums)
barplot_plot <- ggplot(barplot_data, aes(x = Row, y = Sum)) +
geom_bar(stat = "identity", fill = "green") +
labs(title = "Sums of Rows in Matrix B", x = "Row", y = "Sum")
print(barplot_plot)
```