Name: Prateek P

USN: 1MS22CI050

R Lab Programs

Program 1

```
# Step 1: Define a list of products
products <- list(
  list(name = "Apple", price = 0.5),
  list(name = "Banana", price = 0.3),
  list(name = "Milk", price = 2.0),
  list(name = "Bread", price = 1.5),
  list(name = "Eggs", price = 2.5)
)

# Step 2: Initialize shopping cart
shopping_cart <- list()

# Step 3: Define items to be added to the cart
cart_items_to_add <- list(
  list(name = "Apple", quantity = 3),
  list(name = "Milk", quantity = 2)
)

# Step 4: Add items to the shopping cart
for (item in cart_items_to_add) {
  product_name <- item$name
  quantity <- item$quantity

  # Find the product in the list
  product <- NULL
  for (p in products) {
    if (p$name == product_name) {
      product <- p
      break
    }
  }

  # Check if product was found
  if (!is.null(product)) {
    cart_item <- list(name = product$name, price = product$price, quantity =
quantity)
    shopping_cart <- append(shopping_cart, list(cart_item))
    cat("Item added to cart.\n")
  } else {
    cat("Product not found.\n")
  }
}
```

```
# Step 5: Calculate and display receipt
subtotal <- 0
cat("\nReceipt:\n")
for (item in shopping_cart) {
  item_subtotal <- item$price * item$quantity
  cat(item$name, "(", item$quantity, "units) - Price: $", item$price, " -
Subtotal: $", item_subtotal, "\n")
  subtotal <- subtotal + item_subtotal
}

# Step 6: Calculate tax and total cost
tax_rate <- 0.08
tax_amount <- subtotal * tax_rate
total_cost_before_tax <- subtotal
total_cost <- total_cost_before_tax + tax_amount

# Step 7: Display totals
cat("\nSubtotal: $", total_cost_before_tax, "\n")
cat("Tax Amount (8%): $", tax_amount, "\n")
cat("Total Cost: $", total_cost, "\n")
```

Program 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.

1. Declare constants for the number of students (`num_students`) and the number of courses (`num_courses`).
2. Initialize an empty list to store student information.
3. 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.

```
# Constants
num_students <- 5
num_courses <- 5

# Predefined student names
student_names <- c("Arun Rahul", "Bheem Kumar","Raj jumar","Jahal A R","Suresh")
# Predefined course marks for each student
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)

# Initialize a list to store student information
student_records <- list()

# Loop for each student
for (student_index in 1:num_students)
{
    student_name <- student_names[student_index]
    # Initialize variables for calculations
    total_marks <- sum(course_marks[student_index, ])
    average_marks <- total_marks / num_courses
    # Determine grade based on average marks
    grade <- ifelse(average_marks >= 90, "A",
                    ifelse(average_marks >= 80, "B",
                           ifelse(average_marks >= 70, "C",
                                  ifelse(average_marks >= 60, "D", "F"))))
    # Store student information in a record
    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))
}

# Display student information
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(Program 2)

```
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
Name: Jahal A R
Marks: 78 45 67 89 90
Total Marks: 369
Average Marks: 73.8
Grade: C
Name: Suresh
Marks: 89 80 67 78 90
Total Marks: 404
Average Marks: 80.8
Grade: B
```

Program 3

Question

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.

1. Input the number of days the book is overdue.
2. Use conditional statements to calculate the fine amount based on the library's policy.
3. 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 # No fine for the first 7 days
    } else if (days_overdue <= 30)
    {
        fine_per_day <- 2 # Fine per day after 7 days
        fine <- (days_overdue - 7) * fine_per_day
    }
    else
    {
        fine_cap <- 50 # Maximum fine after 30 days
        fine <- fine_cap
    }
    return(fine)
}

# Input number of days overdue
days_overdue <- as.integer(readline("Enter the number of days the book is overdue:
```

```
"))

# Calculate fine
fine_amount <- calculate_fine(days_overdue)

# Display fine information
cat("Fine Amount:", fine_amount, "\n")

if (fine_amount == 0)
{
    cat("No fine. Thank you for returning the book on time!\n")
}
else
{
    if (days_overdue > 30)
    {
        cat("Fine exceeds the maximum cap. Please contact the library.\n")
    }
    else
    {
        cat("Please pay the fine within the specified period.\n")
    }
}
```

Output(Program 3)

```
Enter the number of days the book is overdue: 12
Fine Amount: 10
Please pay the fine within the specified period.
```

Program 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.

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

```
# Initialize arrays for inventory items and quantities
inventory_items <- character(0)
inventory_quantities <- numeric(0)
```

```r
# Function to add a new item with quantity
add_item <- function(item, quantity) {
  inventory_items <<- c(inventory_items, item)
  inventory_quantities <<- c(inventory_quantities, quantity)
  cat("Item added to inventory.\n")
}

# Function to update quantity of an existing item
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")
  }
}
# Function to display inventory
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]))
  }
}

# Main program
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")
  }
}
```

Program 5 (Dataframes)

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.

1. Create a data frame to store student information with columns: "Name", "Math_Score", "Science_Score", "History_Score", "Attendance".
2. 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.
3. Use the functions to analyze student performance and generate the report.

```
# Load the 'dplyr' package for data manipulation
library(dplyr)

# Create a data frame to store student information
student_data <- data.frame(
  Name = character(0),
  Math_Score = numeric(0),
  Science_Score = numeric(0),
  History_Score = numeric(0),
  Attendance = numeric(0)
)

# Function to add student information
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")
}

# Function to calculate average scores
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)
}

# Function to identify students with low attendance
identify_low_attendance <- function(threshold) {
  low_attendance <- student_data %>%
    filter(Attendance < threshold) %>%
```

```
    select(Name, Attendance)
  return(low_attendance)
}

# Function to generate a performance report
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)
}

# Main program
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")
  }
}
```

Program 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:

1. Load the required libraries, including the `forecast` package.
2. 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).
3. Convert the sales data into a time series object with a monthly frequency.
4. Fit an ARIMA (AutoRegressive Integrated Moving Average) model to the sales time series using the `auto.arima()` function.
5. Forecast sales for the next three months using the fitted ARIMA model and the `forecast()` function.
6. Display the forecasted sales results, including point forecasts and prediction intervals.

```r
# (i) Installing and loading the 'forecast' library
# Install the library
install.packages("forecast")
# Load the library
library(forecast)

# (ii) Creating and printing out the sales data frame with Month and Sales columns
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)
)
# Printing out the "sales" dataframe
print(sales_data)

# (iii) Convert the sales data into a time series object with monthly frequency
# Convert to time series
sales_ts <- ts(sales_data$Sales, frequency = 12)
# Printing out converted "sales" time series
print(sales_ts)

# Checking the trend of the sales data
plot(sales_ts)

# Checking the auto correlation factor of the sales data
acf_result <- acf(sales_ts)
print(acf_result)

# (iv) Fit an ARIMA model to the sales time series using the auto.arima() function
# Fit ARIMA model
arima_model <- auto.arima(sales_ts)
print(arima_model)

# (v) Forecast the sales for the next 3 months using the fitted ARIMA model and
the forecast() function
# Forecast sales for next 3 months
forecast_result <- forecast(arima_model, h = 3)

# (vi) Display the forecasted sales results, including point forecasts and
prediction intervals
# Display forecasted sales results
print(forecast_result)
```

Program 7 (Customer Purchase Analysis)

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 an 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

Create a histogram to visualize the distribution of purchase amounts using the `ggplot2` package. Display the histogram with appropriate labels and titles.

```
# Load required libraries
library(dplyr)
library(ggplot2)

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

# Calculate statistical measures
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)

# Display results
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")

# Create a histogram
ggplot(purchase_data, aes(x = PurchaseAmount)) +
  geom_histogram(binwidth = 50, fill = "blue", color = "black") +
```

```
  labs(title = "Distribution of Purchase Amounts", x = "Purchase Amount", y =
"Frequency")
```

Program 8 (Matrix Manipulation)

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.

```
# Task 1: Matrix Creation
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)

# Task 2: Matrix Manipulation
sum_matrix <- matrix_A + matrix_B
scaled_matrix <- matrix_A * 2

# Task 3: Matrix Operations
transposed_A <- t(matrix_A)
product_matrix <- matrix_A %*% matrix_B

# Task 4: Matrix Statistics
sum_matrix_A <- sum(matrix_A)
mean_matrix_B <- mean(matrix_B)
sd_matrix_B <- sd(matrix_B)

# Task 5: Visualization
library(ggplot2)
library(reshape2)

# Create a heatmap of matrix_A
heatmap_data <- melt(matrix_A)
heatmap_plot <- ggplot(heatmap_data, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Heatmap of Matrix A", x = "Column", y = "Row")

# Create a bar plot comparing sums of rows in matrix_B
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")

# Display the visualizations
print(heatmap_plot)
print(barplot_plot)
```