Part A

1A. Suppose that queue <- c ("Steve", "Russell", "Alison", "Liam") and that queue represents a supermarket queue with Steve first in line. Using R expressions update the supermarket queue as successively:

a. **Barry arrives**
b. **Steve is served**
c. **Pam takes her way to the front with one item**
d. **Barry gets impatient and leaves**
e. **Alison gets impatient and leaves**

For the last case you should not assume that you know where in the queue Alison is standing.

Finally, using the function which(x), find the position of Russell in the queue

```
#set queue
queue <- c("Steve", "Russell", "Alison", "Liam") #add barry to the end of the queue queue[length(queue)+1]="Barry"
#remove steve from the queue queue <- queue[queue != "Steve"]
#add pam to the beginning of the queue queue <- append(queue, "Pam", after = 0) #remove barry from queue
queue <- queue[queue != "Barry"] #remove alison from queue
queue <- queue[queue != "Alison"] queue
#find russell's position in the queue which(queue=="Russell")
```

2A. Plot the graph of linear equation, Non-linear equation : square, cubic, square root, etc.,

```
x <- seq(-5, 10, by = 0.1)
M <- 2
C <- 4
eq <- M * x + C
neq1 <- x^2
neq2 <- x^3
neq3 <- sqrt(abs(x))
plot(x,eq, type = "l", col = "blue ", lty = 2,xlab = "X", ylab = "Y", main = "Graphs of Linear and Non-linear Equations")
lines(x, neq1, col = "red")
lines(x, neq2, col = "green")
lines(x, neq3, col = "purple")
legend("topright", legend = c("Linear (Y = 2X + 5)", "Non-linear (Y = X^2)", "Non-linear (Y = X^3)", "Non-linear (Y = √X)"),  col = c("blue", "red", "green", "purple"), lty = c(2, 1, 1, 1), cex = 0.8)
```

3A. The performance of a student in 3rd semester CSE as given below {SUB1, SUB2, SUB3, SUB4, SUB5, SUB6} with score {91, 73, 65, 45, 54, 32}. Draw the Bar chart to indicate the performance of the student Draw the pie chart to indicate the performance of the student Find the average marks and discuss qualitatively about the performance in each subject.

```
subject<- c("MFC", "DAA", "MCES", "OS", "BFE", "UHV")
marks <- c(91, 73, 65, 45, 54, 32)
avg<- mean(marks)
barplot(marks, names.arg = subject, col = "skyblue",main = "Student's Performance i n
3rd       Semester CSE", xlab = "Subjects", ylab = "Marks ")
pie(marks, labels = subject, col = rainbow(length(marks)), main = "Student's Performance
in 3 rd Semester CSE")
cat("Average Marks:", avg, "\n")
cat("Qualitative Performance:\n")
for (i in 1:length(marks)) {
  cat(subject[i], ":", ifelse(marks[i] >= avg, "Good", "Needs Improvement"), "\n")
}
```

4A. Write a R program to extract first 10 English letters in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case.

```
print("First 10 letters in lower case")

t=head(letters,10)
print(t)
print("Last 10 letters in uppercase")
t=tail(LETTERS,10)
print(t)
print ("Letters between 20 to 24")
print(tail(LETTERS[20:24]))
```

5A Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

```
print("Two vectors of different lengths:")
v1 = c(1,3,4,5)
v2 = c(10,11,12,13,14,15)
print(v1)
print(v2)
result = array(c(v1,v2),dim = c(3,3,2))
print("New array:")
print(result)
print("The second row of the second matrix of the array:")
print(result[2, ,2])
print("The element in the 3rd row and 3rd column of the 1st matrix:")
print(result[3,3,1])
```

Part B

1B. 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.**

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

```
# List of products
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)
)
# Available Products
print("The available Products are:")
print(products)
# Initialize shopping cart as an empty list
shopping_cart <- list()
# Define items to be added to the cart
cart_items_to_add <- list (
  list(name = "Apple", quantity = 3),
  list(name = "Milk", quantity = 2) )
# 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
    }   }
  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")
  } }
# Calculate and display receipt
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)
```

2. **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.**

Program:

```
# 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")
}
```

3B. 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.**

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

4B. 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.

```
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)
)
print(student_data)
# 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")
  }
}
```

5B. 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.

```
# Load required libraries
library('forecast')
# Create a 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)
)
print(sales_data)
# Convert to time series
sales_ts <- ts(sales_data$Sales, frequency = 12)
# Fit ARIMA model
arima_model <- auto.arima(sales_ts)
# Forecast sales for next 3 months
forecast_result <- forecast(arima_model, h = 3)
# Display forecast results
print(forecast_result)
```