1. Using R programming, Create a vector containing the ages of 5 people. Find the mean and median age.

   Code:

```
# Creating a vector containing the ages of 5 people
ages <- c(25, 30, 35, 40, 45)

# Finding the mean age
mean_age <- mean(ages)

# Finding the median age
median_age <- median(ages)

# Displaying the mean and median age
print(paste("Mean age:", mean_age))
print(paste("Median age:", median_age))
```

2. Using R programming, Create a vector of 10 random numbers between 1 and 100. Calculate the sum of even numbers.

   Code:

```
#  Generating a vector of 10 random numbers between 1 and 100
random_numbers <- sample(1:100, 10)

# To identify even numbers in the vector
even_numbers <- random_numbers[random_numbers %% 2 == 0]

# To calculate the sum of even numbers
sum_even <- sum(even_numbers)

# Displaying the results
print("Generated Random Numbers:")
print(random_numbers)
print("Even Numbers:")
print(even_numbers)
print("Sum of Even Numbers:")
print(sum_even)
```

3. Using R programming, Create a vector of your friend's names. Print the names in reverse order and then sort the vector.

   Code:

```
# Prompt to enter friend names separated by commas
input_names <- readline(prompt = "Enter your friend's names separated by commas: ")

# Splitting the input string into a vector of names
friend_names <- strsplit(input_names, ",")[[1]]

# Printing the names in reverse order
print(rev(friend_names))

# Sorting the vector of names
sorted_names <- sort(friend_names)

# Printing the sorted vector of names
print(sorted_names)
```

4. Using R programming, find the position of the alphabet in the English letters.

   Code:

```
# Function to find the position of an alphabet in the English letters
find_alphabet_position <- function(alphabet) {
  # To convert alphabet to lowercase to handle case-insensitivity
  alphabet <- tolower(alphabet)

  # defining the English alphabet
  english_alphabet <- letters

  # Finding the position of the alphabet
  position <- which(english_alphabet == alphabet)

  # If alphabet is found, return its position, otherwise return a message
  if (length(position) > 0) {
    return(position)
  } else {
    return(paste("Alphabet", alphabet, "not found in English letters."))
  }
}

# to get input from the user
alphabet_input <- readline(prompt = "Enter an alphabet: ")

# Calling the function to find the position of the input alphabet
position <- find_alphabet_position(alphabet_input)

# Print the position
print(position)
```

5. Create a data frame with the columns as height, weight and age. Convert this data frame into a matrix.

   Code:

```
# Creating a data frame with height, weight, and age
data <- data.frame(
  height = c(170, 165, 180, 175),
  weight = c(70, 65, 80, 75),
  age = c(25, 30, 28, 35)
)

# Printing the data frame
print("Data Frame:")
print(data)

# Converting the data frame into a matrix
matrix_data <- as.matrix(data)

# Print the matrix
print("Matrix:")
print(matrix_data)
```

6. Write a program to print the multiplication table of user's choice.

   Code:

```
# Function to print the multiplication table
print_multiplication_table <- function(number) {
  cat("Multiplication Table of", number, ":\n")
  for (i in 1:10) {
    cat(number, "x", i, "=", number * i, "\n")
  }
}

# Getting input from the user
number <- as.numeric(readline(prompt = "Enter a number: "))

# Checks if the input is a number
if (!is.na(number)) {
  # Calls the function to print the multiplication table
  print_multiplication_table(number)
} else {
  print("Invalid input. Please enter a valid number.")
}
```

7. Write a program to print the following pattern.

```
1
1       2
1       2       3
1       2       3       4
```

Code:

```
# Function to print the pattern
print_pattern <- function(rows) {
  for (i in 1:rows) {
    for (j in 1:i) {
      cat(j, "\t")
    }
    cat("\n")
  }
}

# To get input from the user at runtime
rows <- as.integer(readline(prompt = "Enter the number of rows for the pattern: "))

# Checks if the input is a positive integer
if (!is.na(rows) && rows > 0) {
  # Calling the function to print the pattern
  print_pattern(rows)
} else {
  print("Invalid input. Please enter a positive integer.")
}
```

8. Write a program that reads the ToothGrowth data set.
    a. Add a new column to the data set named condition. if the length is less than 10, then the corresponding value of condition should be All Well, and caution otherwise.
    b. Find the frequency of each condition

Code:

```
# Reading the ToothGrowth dataset
data <- ToothGrowth

# Adding a new column named "condition"
data$condition <- ifelse(data$len < 10, "All Well", "Caution")

# To Print the updated dataset
print(data)


# Finding the frequency of each condition
frequency <- table(data$condition)

# Printing the frequency of each condition
print(frequency)
```

9. Write a program that reads the ChickWeight dataset.
    a. Add a new column named weight_gain_rate to it containing the average weight gain per day for each chick.
    b. If the weight gain per day is greater than 10 grams, set the corresponding value of weight_gain to High; otherwise, set it to Low.
    c. Find the Frequency of each levels in the weight_gain column.

Code:

```
# Reading the ChickWeight dataset
data <- ChickWeight

# Calculating average weight gain per day for each chick
data$weight_gain_rate <- data$weight / data$Time

# Setting weight_gain_rate based on the condition
data$weight_gain_rate <- ifelse(data$weight_gain_rate > 10, "High", "Low")

# printing the first 6 rows of data
print(head(data))

# Finding the Frequency of each level in the weight_gain_rate column
frequency <- table(data$weight_gain_rate)

# Printing the Frequency of each level in the weight_gain_rate column
print(frequency)
```

9. Convert a vector of test scores (90, 85, 75, 80, 95) into a factor with levels "Excellent" (scores >= 90), "Good" (80 <= scores < 90), "Average" (70 <= scores < 80), and "Below Average" (scores < 70).

Code:

```
# Defining the vector of test scores
test_scores <- c(90, 85, 75, 80, 95)

# Creating an empty vector to store the levels
levels_vector <- character(length(test_scores))

# Looping through each test score
for (i in seq_along(test_scores)) {
  # Assign levels based on conditions
  if (test_scores[i] >= 90) {
    levels_vector[i] <- "Excellent"
  } else if (test_scores[i] >= 80) {
    levels_vector[i] <- "Good"
  } else if (test_scores[i] >= 70) {
    levels_vector[i] <- "Average"
  } else {
    levels_vector[i] <- "Below Average"
  }
}
```

```
# Converting the levels vector to a factor
factor_scores <- factor(levels_vector, levels = c("Below Average", "Average", "Good",
"Excellent"))

# Printing the factor scores
print(factor_scores)

# Printing the frequency of each level
print(table(factor_scores))
```

10. Create a 3x3 matrix with random integers between 1 and 15.
    a. Extract a row or column from a matrix.
    b. Replace a row or column in a matrix with new values.

Code:

```
# Function to create a matrix with random integers between 1 and 15
create_random_matrix <- function(rows, cols) {
  matrix(sample(1:15, rows * cols, replace = TRUE), nrow = rows, ncol = cols)
}

# Function to print the matrix
print_matrix <- function(matrix) {
  print(matrix)
}

# Function to extract a row from the matrix
extract_row <- function(matrix, row_index) {
  row <- matrix[row_index, ]
  print(row)
}

# Function to extract a column from the matrix
extract_column <- function(matrix, col_index) {
  column <- matrix[, col_index]
  print(column)
}

# Function to replace a row in the matrix with new values
replace_row <- function(matrix, row_index, new_values) {
  matrix[row_index, ] <- new_values
  print(matrix)
}

# Function to replace a column in the matrix with new values
replace_column <- function(matrix, col_index, new_values) {
  matrix[, col_index] <- new_values
  print(matrix)
}
```

```
# Main program
# Create a 3x3 matrix with random integers between 1 and 15
matrix <- create_random_matrix(3, 3)

# Print the matrix
print_matrix(matrix)

# Ask the user for their choice
choice <- as.integer(readline(prompt = "Enter 1 to extract a row, 2 to extract a column, 3 to
replace a row, or 4 to replace a column: "))

# Perform the selected operation based on user's choice
if (choice == 1) {
  row_index <- as.integer(readline(prompt = "Enter the row index to extract: "))
  extract_row(matrix, row_index)
} else if (choice == 2) {
  col_index <- as.integer(readline(prompt = "Enter the column index to extract: "))
  extract_column(matrix, col_index)
} else if (choice == 3) {
  row_index <- as.integer(readline(prompt = "Enter the row index to replace: "))
  new_values <- as.integer(readline(prompt = "Enter new values for the row separated by space:
"))
  replace_row(matrix, row_index, new_values)
} else if (choice == 4) {
  col_index <- as.integer(readline(prompt = "Enter the column index to replace: "))
  new_values <- as.integer(readline(prompt = "Enter new values for the column separated by
space: "))
  replace_column(matrix, col_index, new_values)
} else {
  print("Invalid choice.")
}
```