```r
# Part A: Arithmetic Operations and Variable Assignments

  # 1. Calculator Operations:

      # Add, subtract, multiply, and divide any two numbers.

      # Addition
      num1 <- 5
      num2 <- 3
      sum <- num1+num2
      print(paste("Addition : ",sum))
      # Addition : 8

      # Subtraction
      num1 <- 2
      num2 <- 6
      sub <- num1-num2
      print(paste("Subtraction : ",sub))
      # Subtraction : -4

      # Multiplication
      num1 <- 4
      num2 <- 6
      mult <- num1*num2
      print(paste("Multiplication : ",mult))
      # Multiplication : 24

      # Division
      num1 <- 9
      num2 <- 3
      div <- num1/num2
      print(paste("Division : ",div))
      # Division : 3

      # Compute 8^2, the logarithm of 256 (base 2), and the square root of 121.

      # Power
      num1 <- 8
      num2 <- 2
      pow <- num1^num2
      print(paste("Power : ",pow))
      # Power : 64

      # Logarithm
      num1 <- 256
      num2 <- 2
      log <- log(num1, base = num2)
      print(paste("Logarithm : ",log))
      # Logarithm : 8

      # Square Root
      num1 <- 121
      sqrt <- sqrt(num1)
      print(paste("Square Root : ",sqrt))
      # Square Root : 11

      # Find the sine, cosine, and tangent of 30°, 45°, and 60° (ensure conversion to
radians).

      angles <- c(30, 45, 60)
      radians <- angles * pi / 180

      sine_val <- sin(radians)
      cosine_val <- cos(radians)
```

```
      tangent_val <- tan(radians)
      print(data.frame(Angle = angles, Sine = sine_val, Cosine = cosine_val, Tangent =
tangent_val))

      # output:
      # Angle     Sine      Cosine    Tangent
      #  30    0.5000000  0.8660254  0.5773503
      #  45    0.7071068  0.7071068  1.0000000
      #  60    0.8660254  0.5000000  1.7320508

    # 2. Variable Assignments:

      # Assign x = 50 and y = 30, and compute x+y, x-y, x*y, x/y, x^2 - y^2.

      x <- 50
      y <- 30

      sum_xy <- x + y
      sub_xy <- x - y
      mult_xy <- x * y
      div_xy <- x / y
      exp_xy <- x^2 - y^2

      print(paste("x + y =", sum_xy))
      print(paste("x - y =", sub_xy))
      print(paste("x * y =", mult_xy))
      print(paste("x / y =", div_xy))
      print(paste("x^2 - y^2 =", exp_xy))

      # Output
      # "x + y = 80"
      # "x - y = 20"
      # "x * y = 1500"
      # "x / y = 1.66666666666667"
      # "x^2 - y^2 = 1600"

      # Assign z = 100 and calculate its square root and exponential value.

      z <- 100

      sqrt_z <- sqrt(z)
      exp_z <- exp(z)

      print(paste("Square root of z =", sqrt_z))
      print(paste("Exponential of z =", exp_z))

      # Output
      # "Square root of z = 10"
      # "Exponential of z = 2.68811714181614e+43"

# Part B: Built-in Functions and Data Types

    # 3. Data Types:

      # Create a numeric variable, a character string, and a logical value.

      num_var <- 42
      char_var <- "Hello"
      log_var <- TRUE

      # Use built-in functions to check their data types.

      print(paste("Type of num_var:", typeof(num_var)))
      print(paste("Class of num_var:", class(num_var)))
```

```r
  print(paste("Type of char_var:", typeof(char_var)))
  print(paste("Class of char_var:", class(char_var)))

  print(paste("Type of log_var:", typeof(log_var)))
  print(paste("Class of log_var:", class(log_var)))

  # Output
  # "Type of num_var: double"
  # "Class of num_var: numeric"
  # "Type of char_var: character"
  # "Class of char_var: character"
  # "Type of log_var: logical"
  # "Class of log_var: logical"

# 4. Built-in Functions:

  # Mathematical: abs, round, ceiling, floor.

  num <- -12.7

  abs_val <- abs(num)
  round_val <- round(num)
  ceil_val <- ceiling(num)
  floor_val <- floor(num)

  print(paste("Absolute value:", abs_val))
  print(paste("Rounded value:", round_val))
  print(paste("Ceiling value:", ceil_val))
  print(paste("Floor value:", floor_val))

  # Output
  # "Absolute value: 12.7"
  # "Rounded value: -13"
  # "Ceiling value: -12"
  # "Floor value: -13"

  # Statistical: mean, median, sum, sd.

  data_vec <- c(10, 20, 30, 40, 50)

  mean_val <- mean(data_vec)
  median_val <- median(data_vec)
  sum_val <- sum(data_vec)
  sd_val <- sd(data_vec)

  print(paste("Mean:", mean_val))
  print(paste("Median:", median_val))
  print(paste("Sum:", sum_val))
  print(paste("Standard Deviation:", sd_val))

  # Output
  # "Mean: 30"
  # "Median: 30"
  # "Sum: 150"
  # "Standard Deviation: 15.8113883008419"

  # Sequence Generation: seq, rep.

  seq_vec <- seq(1, 10, by = 2)
  rep_vec <- rep(5, times = 4)

  print("Sequence:")
  print(seq_vec)
  print("Repeated values:")
  print(rep_vec)
```

```
      # Output
      # "Sequence:"
      # 1 3 5 7 9
      # "Repeated values:"
      # 5 5 5 5

      # sCharacter Functions: toupper, tolower, nchar.

      char_str <- "Hello World !"

      upper_str <- toupper(char_str)
      lower_str <- tolower(char_str)
      char_length <- nchar(char_str)

      print(paste("Uppercase:", upper_str))
      print(paste("Lowercase:", lower_str))
      print(paste("Character count:", char_length))

      # Output
      # "Uppercase: HELLO WORLD !"
      # "Lowercase: hello world !"
      # "Character count: 13"

# Part C: Data Structures and Subsetting

    # 5. Vectors:

      # Create a vector of integers from 1 to 15.

      vec <- 1:15
      print("Original vector:")
      print(vec)

      # Output
      # "Original vector:"
      # 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

      # Add 3 to each element.

      vec_modified <- vec + 3
      print("Vector after adding 3:")
      print(vec_modified)

      # Output
      # "Vector after adding 3:"
      # 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18

      # Extract the 5th, 10th, and last elements.

      extracted_values <- vec_modified[c(5, 10, length(vec_modified))]
      print("Extracted elements (5th, 10th, last):")
      print(extracted_values)

      #Output
      # "Extracted elements (5th, 10th, last):"
      # 8 13 18

      # Calculate the sum, mean, and length of the vector.

      sum_vec <- sum(vec_modified)
      mean_vec <- mean(vec_modified)
      length_vec <- length(vec_modified)

      # Output
```

```
  # "Extracted elements (5th, 10th, last):"
  # 8 13 18

# 6. Matrices:

  # Create a 3x3 matrix with values from 1 to 9.

  matrix_3x3 <- matrix(1:9, nrow = 3, ncol = 3)
  print("3x3 Matrix:")
  print(matrix_3x3)

  # Output
  # "3x3 Matrix:"
  #      [,1] [,2] [,3]
  # [1,]    1    4    7
  # [2,]    2    5    8
  # [3,]    3    6    9

  # Extract the second row, the first column, and the element at (3,3).

  second_row <- matrix_3x3[2, ]
  first_column <- matrix_3x3[, 1]
  element_3_3 <- matrix_3x3[3, 3]

  print("Second row:")
  print(second_row)
  print("First column:")
  print(first_column)
  print("Element at (3,3):")
  print(element_3_3)

  # Output
  # "Second row:"
  # 2 5 8
  # "First column:"
  # 1 2 3
  # "Element at (3,3):"
  # 9

  # Compute the sum of all elements in the matrix.

  sum_matrix_3x3 <- sum(matrix_3x3)
  print("Sum of all elements in 3x3 matrix:")
  print(sum_matrix_3x3)

  # Output
  # "Sum of all elements in 3x3 matrix:"
  # 45

  # Create a 4x4 matrix with row-wise data.

  matrix_4x4 <- matrix(1:16, nrow = 4, ncol = 4, byrow = TRUE)
  print("4x4 Matrix:")
  print(matrix_4x4)

  # Output
  # "4x4 Matrix:"
  #      [,1] [,2] [,3] [,4]
  # [1,]    1    2    3    4
  # [2,]    5    6    7    8
  # [3,]    9   10   11   12
  # [4,]   13   14   15   16

  # Extract the second row, the fourth column, and the element at (4,3).
```

```r
second_row_4x4 <- matrix_4x4[2, ]
fourth_column <- matrix_4x4[, 4]
element_4_3 <- matrix_4x4[4, 3]

print("Second row:")
print(second_row_4x4)
print("Fourth column:")
print(fourth_column)
print("Element at (4,3):")
print(element_4_3)

# Output
# "Second row:"
# 5 6 7 8
# "Fourth column:"
# 4 8 12 16
# "Element at (4,3):"
# 15

# Compute the sum of all elements in the matrix.

sum_matrix_4x4 <- sum(matrix_4x4)
print("Sum of all elements in 4x4 matrix:")
print(sum_matrix_4x4)

# Output
# "Sum of all elements in 4x4 matrix:"
# 136

# 7. Lists:

# Create a list containing:
# - A string: 'Learning R'
# - A numeric value: 3.14159
# - A vector: c(10, 20, 30, 40, 50)

my_list <- list(
  text = "Learning R",
  number = 3.14159,
  vector = c(10, 20, 30, 40, 50)
)

print("Created List:")
print(my_list)

# Output
# "Created List:"
# $text
# [1] "Learning R"

# $number
# [1] 3.14159

# $vector
# [1] 10 20 30 40 50

# Extract each component of the list individually.

text_element <- my_list$text
number_element <- my_list$number
vector_element <- my_list$vector

print("Extracted text element:")
print(text_element)
```

```r
print("Extracted numeric element:")
print(number_element)

print("Extracted vector element:")
print(vector_element)

# Output
# "Extracted text element:"
# [1] "Learning R"

# "Extracted numeric element:"
# [1] 3.14159

# "Extracted vector element:"
# [1] 10 20 30 40 50

# 8. Data Frames (Computer Science Example):

# Create a data frame with the following columns:
# - Student_ID: A unique identifier for each student.
# - Name: Names of the students.
# - Course: Courses (e.g., Data Science, AI, Cybersecurity).
# - Marks: Marks scored by students out of 100.

students_df <- data.frame(
  Student_ID = 1:5,
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  Course = c("Data Science", "AI", "AI", "Maths", "AI"),
  Marks = c(85, 92, 78, 88, 95)
)

print("Created Data Frame:")
print(students_df)

# Output
# "Created Data Frame:"
# Student_ID  Name       Course Marks
# 1          1 Alice Data Science    85
# 2          2   Bob           AI    92
# 4          4 David        Maths    88
# 5          5   Eva           AI    95

# Perform subsetting to:
# - Extract students enrolled in the 'AI' course.

ai_students <- subset(students_df, Course == "AI")
print("Students enrolled in AI course:")
print(ai_students)

# Output
# "Students enrolled in AI course:"
#   Student_ID   Name Course Marks
# 2          2    Bob     AI    92
# 3          3 Charlie     AI    78
# 5          5     Eva     AI    95

# - Extract students with marks greater than 80.

marks_above_80 <- subset(students_df, Marks > 80)
print("Students with marks greater than 80:")
print(marks_above_80)

# Output
# Student_ID  Name       Course Marks
# 1          1 Alice Data Science    85
```

```
# 2          2   Bob           AI    92
# 4          4 David        Maths    88
# 5          5   Eva           AI    95

# Add a new column, Programming_Language, with entries like R, Python, or Java.

students_df$Programming_Language <- c("R", "Python", "Python", "Java", "R")
print("Updated Data Frame with Programming_Language column:")
print(students_df)

# Output
# "Updated Data Frame with Programming_Language column:"
#   Student_ID    Name       Course Marks Programming_Language
# 1          1   Alice Data Science    85                    R
# 2          2     Bob           AI    92               Python
# 3          3 Charlie           AI    78               Python
# 4          4   David        Maths    88                 Java
# 5          5     Eva           AI    95                    R

# Part D: Advanced Practice

  # 9. Logarithmic and Exponential Functions:

    # Compute the natural logarithm and exponential of values 1, 2, 3.

    values <- c(1, 2, 3)

    # Natural logarithm (log base e)

    log_values <- log(values)
    print("Natural logarithm of values 1, 2, 3:")
    print(log_values)

    # Output
    # "Natural logarithm of values 1, 2, 3:"
    # 0.000000 0.693147 1.098612

    # Exponential (e^x)

    exp_values <- exp(values)
    print("Exponential of values 1, 2, 3:")
    print(exp_values)

    # Output
    # "Exponential of values 1, 2, 3:"
    # 2.718282 7.389056 20.085537

  # 10. Trigonometric Functions:

    # Convert degrees to radians since R trigonometric functions expect radians.

    degrees <- c(15, 45, 90)
    radians <- degrees * (pi / 180)

    # Calculate the sine, cosine, and tangent of angles 15°, 45°, and 90°.

    sine_values <- sin(radians)
    cosine_values <- cos(radians)
    tangent_values <- tan(radians)

    print("Sine values of 15°, 45°, 90°:")
    print(sine_values)

    # Output
    # "Sine values of 15°, 45°, 90°:"
```

```
  # 0.2588190 0.7071068 1.0000000

  print("Cosine values of 15°, 45°, 90°:")
  print(cosine_values)

  # Output
  # "Cosine values of 15°, 45°, 90°:"
  # 0.9659258 0.7071068 0.0000000

  print("Tangent values of 15°, 45°, 90°:")
  print(tangent_values)

  # Output
  # "Tangent values of 15°, 45°, 90°:"
  # 0.2679492 1.0000000 NaN

# 11. Explore Basic Statistical Functions:

  # Create a numeric vector

  num_vector <- c(5, 10, 15, 20, 25)

  # Use length to find the size of a vector.

  vector_length <- length(num_vector)
  print("Size of the vector:")
  print(vector_length)

  # Output
  # "Size of the vector:"
  #  5

  # Use sum to compute the total of elements in a numeric vector.

  vector_sum <- sum(num_vector)
  print("Total sum of elements in the vector:")
  print(vector_sum)

  # Output
  # "Total sum of elements in the vector:"
  #  75

  # Use seq to generate a sequence from 1 to 10 with a step of 2.

  sequence <- seq(1, 10, by = 2)
  print("Generated sequence using seq (1 to 10, step 2):")
  print(sequence)

  # Output
  # "Generated sequence using seq (1 to 10, step 2):"
  # 1 3 5 7 9

  # Use rep to generate repetitions of the number 3, five times.

  repeated_values <- rep(3, 5)
  print("Generated repetitions using rep (3 repeated 5 times):")
  print(repeated_values)

  # Output
  # "Generated repetitions using rep (3 repeated 5 times):"
  # 3 3 3 3 3

# 12. Bonus Practice:

  # Generate a sequence of numbers from 10 to 100 in steps of 10.
```

```
      sequence_numbers <- seq(10, 100, by = 10)
      print("Sequence of numbers from 10 to 100 in steps of 10:")
      print(sequence_numbers)

      # Output
      # "Sequence of numbers from 10 to 100 in steps of 10:"
      # [1]  10  20  30  40  50  60  70  80  90 100

      # Repeat the string 'R Programming' 5 times.

      repeated_string <- rep("R Programming", 5)
      print("Repeated string 'R Programming' 5 times:")
      print(repeated_string)

      # Output
      # "Repeated string 'R Programming' 5 times:"
      # [1] "R Programming" "R Programming" "R Programming" "R Programming" "R
Programming"
```