1. Write a program that prints _Hello World' to the screen.

```
  cat("Hello World\n")
```

Output:

```
Hello World
```

2. Write a program that asks the user for a number n and prints the sum of the numbers 1 to n.

```
 # Get user input for n
n <- as.numeric(readline("Enter a positive integer (n): "))

# Check if n is a positive integer
if (is.numeric(n) && n > 0 && n == round(n)) {
  # Calculate the sum of the numbers from 1 to n
  result <- sum(1:n)

  # Print the result
  cat("The sum of the numbers from 1 to", n, "is:", result, "\n")
} else {
  cat("Please enter a valid positive integer.\n")
}
```

Output:

```
> source("D:/Desktop/Data Analytics/02.R", echo=TRUE)

> # Get user input for n
> n <- as.numeric(readline("Enter a positive integer (n): "))
Enter a positive integer (n): 3

> # Check if n is a positive integer
> if (is.numeric(n) && n > 0 && n == round(n)) {
+   # Calculate the sum of the numbers from 1 to n
+   result <- .... [TRUNCATED]
The sum of the numbers from 1 to 3 is: 6
```

3. Write a program that prints a multiplication table for numbers up to 12.

```
# Get user input for the number
num <- as.numeric(readline("Enter a number to display its multiplication table up to 12:
"))

# Check if num is a valid number
if (is.numeric(num)) {
  cat("Multiplication table for", num, "up to 12:\n")

  # Generate and print the multiplication table
  for (i in 1:12) {
    result <- num * i
    cat(num, "x", i, "=", result, "\n")
  }
```

```
} else {
  cat("Please enter a valid number.\n")
}
```

Output:

```
> source("D:/Desktop/Data Analytics/03.R", echo=TRUE)

> # Get user input for the number
> num <- as.numeric(readline("Enter a number to display its multiplication table up to
12: "))
Enter a number to display its multiplication table up to 12: 12

> # Check if num is a valid number
> if (is.numeric(num)) {
+   cat("Multiplication table for", num, "up to 12:\n")
+
+   # Generate and print the  .... [TRUNCATED]
Multiplication table for 12 up to 12:
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132
12 x 12 = 144
```

4. Write a function that returns the largest element in a list.

```
find_largest_element <- function(my_list) {
  if (length(my_list) == 0) {
    stop("Input list is empty.")
  }

  max_element <- max(my_list)
  return(max_element)
}

# Example usage:
my_list <- c(5, 8, 2, 10, 3)
result <- find_largest_element(my_list)
cat("The largest element in the list is:", result, "\n")
```

Output:

```
> source("D:/Desktop/Data Analytics/04.R", echo=TRUE)

> find_largest_element <- function(my_list) {
+   if (length(my_list) == 0) {
+     stop("Input list is empty.")
+   }
+
+   max_element <- max(my_ .... [TRUNCATED]

> # Example usage:
> my_list <- c(5, 8, 2, 10, 3)

> result <- find_largest_element(my_list)
```

```
> cat("The largest element in the list is:", result, "\n")
The largest element in the list is: 10
```

5. Write a function that computes the running total of a list.

```
compute_running_total <- function(my_list) {
  if (length(my_list) == 0) {
    stop("Input list is empty.")
  }

  running_total <- cumsum(my_list)
  return(running_total)
}

# Example usage:
my_list <- c(5, 8, 2, 10, 3)
result <- compute_running_total(my_list)
cat("Running total of the list is:", result, "\n")
```

Output:

```
> source("D:/Desktop/Data Analytics/05.R", echo=TRUE)

> compute_running_total <- function(my_list) {
+   if (length(my_list) == 0) {
+     stop("Input list is empty.")
+   }
+
+   running_total <- cums .... [TRUNCATED]

> # Example usage:
> my_list <- c(5, 8, 2, 10, 3)

> result <- compute_running_total(my_list)

> cat("Running total of the list is:", result, "\n")
Running total of the list is: 5 13 15 25 28
```

6. Write a function that tests whether a string is a palindrome.

```
# Install and load the stringi package
if (!requireNamespace("stringi", quietly = TRUE)) {
  install.packages("stringi")
}
library(stringi)

is_palindrome <- function(input_str) {
  # Convert the string to lowercase to make the comparison case-insensitive
  cleaned_str <- tolower(input_str)

  # Remove spaces and non-alphanumeric characters from the string
  cleaned_str <- gsub("[^a-z0-9]", "", cleaned_str)

  # Check if the cleaned string is equal to its reverse
  return(cleaned_str == stri_reverse(cleaned_str))
}

# Get user input for the string
user_input <- readline("Enter a string to check if it's a palindrome: ")

# Call the function with user input
result <- is_palindrome(user_input)
```

```
# Display the result
if (result) {
  cat("Yes, '", user_input, "' is a palindrome.\n")
} else {
  cat("No, '", user_input, "' is not a palindrome.\n")
}
```

Output:

```
> source("D:/Desktop/Data Analytics/06.R", echo=TRUE)

> # Install and load the stringi package
> if (!requireNamespace("stringi", quietly = TRUE)) {
+   install.packages("stringi")
+ }

> library(stringi)

> is_palindrome <- function(input_str) {
+   # Convert the string to lowercase to make the comparison case-insensitive
+   cleaned_str <- tolower(inpu .... [TRUNCATED]

> # Get user input for the string
> user_input <- readline("Enter a string to check if it's a palindrome: ")
Enter a string to check if it's a palindrome: saurav kr just be dead

> # Call the function with user input
> result <- is_palindrome(user_input)

> # Display the result
> if (result) {
+   cat("Yes, '", user_input, "' is a palindrome.\n")
+ } else {
+   cat("No, '", user_input, "' is not a palin ..." ... [TRUNCATED]
No, ' saurav kr just be dead ' is not a palindrome.
```

7. Implement linear search.

```
linear_search <- function(search_list, target) {
  for (i in seq_along(search_list)) {
    if (search_list[i] == target) {
      return(i)  # Return the index if the target is found
    }
  }
  return(-1)  # Return -1 if the target is not found
}

# Example usage:
my_list <- c(5, 8, 2, 10, 3)
target_value <- 10

result <- linear_search(my_list, target_value)

if (result != -1) {
  cat("The target value", target_value, "was found at index", result, "\n")
} else {
  cat("The target value", target_value, "was not found in the list.\n")
}
```

output:

```
> source("D:/Desktop/Data Analytics/07.R", echo=TRUE)

> linear_search <- function(search_list, target) {
```

```
+   for (i in seq_along(search_list)) {
+     if (search_list[i] == target) {
+       return(i)  #  .... [TRUNCATED]

> # Example usage:
> my_list <- c(5, 8, 2, 10, 3)

> target_value <- 10

> result <- linear_search(my_list, target_value)

> if (result != -1) {
+   cat("The target value", target_value, "was found at index", result, "\n")
+ } else {
+   cat("The target value", target_valu .... [TRUNCATED]
The target value 10 was found at index 4
```

8. Implement binary search.

```
binary_search <- function(sorted_list, target) {
  low <- 1
  high <- length(sorted_list)

  while (low <= high) {
    mid <- floor((low + high) / 2)

    if (sorted_list[mid] == target) {
      return(mid)  # Return the index if the target is found
    } else if (sorted_list[mid] < target) {
      low <- mid + 1
    } else {
      high <- mid - 1
    }
  }

  return(-1)  # Return -1 if the target is not found
}

# Example usage:
sorted_list <- c(2, 3, 5, 8, 10)
target_value <- 8

result <- binary_search(sorted_list, target_value)

if (result != -1) {
  cat("The target value", target_value, "was found at index", result, "\n")
} else {
  cat("The target value", target_value, "was not found in the list.\n")
}
```

output:

```
> source("D:/Desktop/Data Analytics/08.R", echo=TRUE)

> binary_search <- function(sorted_list, target) {
+   low <- 1
+   high <- length(sorted_list)
+
+   while (low <= high) {
+     mid <- floor((low .... [TRUNCATED]

> # Example usage:
> sorted_list <- c(2, 3, 5, 8, 10)

> target_value <- 8
```

```
> result <- binary_search(sorted_list, target_value)

> if (result != -1) {
+   cat("The target value", target_value, "was found at index", result, "\n")
+ } else {
+   cat("The target value", target_valu .... [TRUNCATED]
The target value 8 was found at index 4
>
User 2's Shared Secret Key: 4
```

9. Implement matrix addition, subtraction and Multiplication.

```
# Function to add two matrices
matrix_addition <- function(mat1, mat2) {
  if (!all(dim(mat1) == dim(mat2))) {
    stop("Matrices must have the same dimensions for addition.")
  }

  result <- mat1 + mat2
  return(result)
}

# Function to subtract two matrices
matrix_subtraction <- function(mat1, mat2) {
  if (!all(dim(mat1) == dim(mat2))) {
    stop("Matrices must have the same dimensions for subtraction.")
  }

  result <- mat1 - mat2
  return(result)
}

# Function to multiply two matrices
matrix_multiplication <- function(mat1, mat2) {
  if (ncol(mat1) != nrow(mat2)) {
    stop("Number of columns in the first matrix must be equal to the number of rows in
the second matrix for multiplication.")
  }

  result <- mat1 %*% mat2
  return(result)
}

# Example usage:
matrix1 <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = TRUE)
matrix2 <- matrix(c(5, 6, 7, 8), nrow = 2, byrow = TRUE)

# Matrix addition
result_addition <- matrix_addition(matrix1, matrix2)
cat("Matrix Addition Result:\n", result_addition, "\n\n")

# Matrix subtraction
result_subtraction <- matrix_subtraction(matrix1, matrix2)
cat("Matrix Subtraction Result:\n", result_subtraction, "\n\n")

# Matrix multiplication
result_multiplication <- matrix_multiplication(matrix1, matrix2)
cat("Matrix Multiplication Result:\n", result_multiplication, "\n")
```

output:

```
> source("D:/Desktop/Data Analytics/09.R", echo=TRUE)

> # Matrix multiplication
```

```
> result_multiplication <- matrix_multiplication(matrix1, matrix2)

> cat("Matrix Multiplication Result:\n", result_multiplication, "\n")
Matrix Multiplication Result:
 19 43 22 50
```

10. Fifteen students were enrolled in a course. There ages were:
20 20 20 20 20 21 21 21 22 22 22 22 23 23 23
i) Find the median age of all students under 22 years
ii) Find the median age of all students
iii) Find the mean age of all students
iv) Find the modal age for all students
v) Two more students enter the class. The age of both students is 23. What is now mean, mode and median ?

```
# Ages of the initial fifteen students
ages <- c(20, 20, 20, 20, 20, 21, 21, 21, 22, 22, 22, 22, 23, 23, 23)

# Ages of the additional two students
additional_ages <- c(23, 23)

# Combine the ages of all students
all_ages <- c(ages, additional_ages)

# 1. Find the median age of all students under 22 years
median_under_22 <- median(all_ages[all_ages < 22])

# 2. Find the median age of all students
median_all <- median(all_ages)

# 3. Find the mean age of all students
mean_all <- mean(all_ages)

# 4. Find the modal age for all students
modal_all <- as.numeric(names(table(all_ages))[which.max(table(all_ages))])

# Display results for the initial fifteen students
cat("1. Median age of students under 22:", median_under_22, "\n")
cat("2. Median age of all students:", median_all, "\n")
cat("3. Mean age of all students:", mean_all, "\n")
cat("4. Modal age for all students:", modal_all, "\n")

# Ages of the additional two students
additional_ages <- c(23, 23)

# Combine the ages of all students, including the additional two students
all_ages <- c(all_ages, additional_ages)

# 5. Find the updated mean, mode, and median with the two additional students
updated_mean <- mean(all_ages)
updated_modal <- as.numeric(names(table(all_ages))[which.max(table(all_ages))])
updated_median <- median(all_ages)

cat("\nAfter two more students enter the class:\n")
cat("   Updated Mean age:", updated_mean, "\n")
cat("   Updated Modal age:", updated_modal, "\n")
cat("   Updated Median age:", updated_median, "\n")
```

output:

```
> source("D:/Desktop/Data Analytics/10.R", echo=TRUE)
```

```
> # Ages of the initial fifteen students
> ages <- c(20, 20, 20, 20, 20, 21, 21, 21, 22, 22, 22, 22, 23, 23, 23)

> # Ages of the additional two students
> additional_ages <- c(23, 23)

> # Combine the ages of all students
> all_ages <- c(ages, additional_ages)

# Display results for the initial fifteen students
> cat("1. Median age of students under 22:", median_under_22, "\n")
1. Median age of students under 22: 20

> cat("2. Median age of all students:", median_all, "\n")
2. Median age of all students: 22

> cat("3. Mean age of all students:", mean_all, "\n")
3. Mean age of all students: 21.52941

> cat("4. Modal age for all students:", modal_all, "\n")
4. Modal age for all students: 20
After two more students enter the class:

> cat("   Updated Mean age:", updated_mean, "\n")
   Updated Mean age: 21.68421

> cat("   Updated Modal age:", updated_modal, "\n")
   Updated Modal age: 23

> cat("   Updated Median age:", updated_median, "\n")
   Updated Median age: 22
```