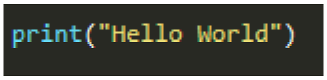
Experiment – 1

Aim: Write a program that prints “Hello World” to the Screen.

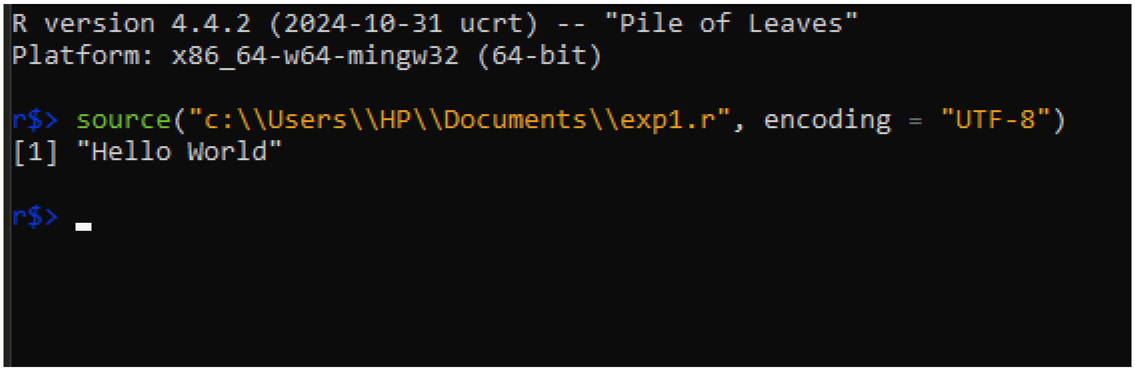
Tools Used: R version 4.4.2

Program:



```r
print("Hello World")
```

Output:



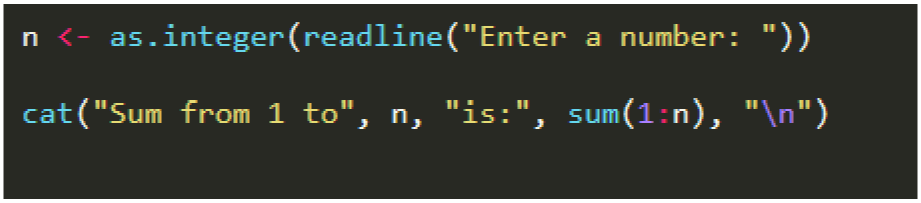
```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("c:\\Users\\HP\\Documents\\exp1.r", encoding = "UTF-8")
[1] "Hello World"

r$> 
```

Experiment – 2

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

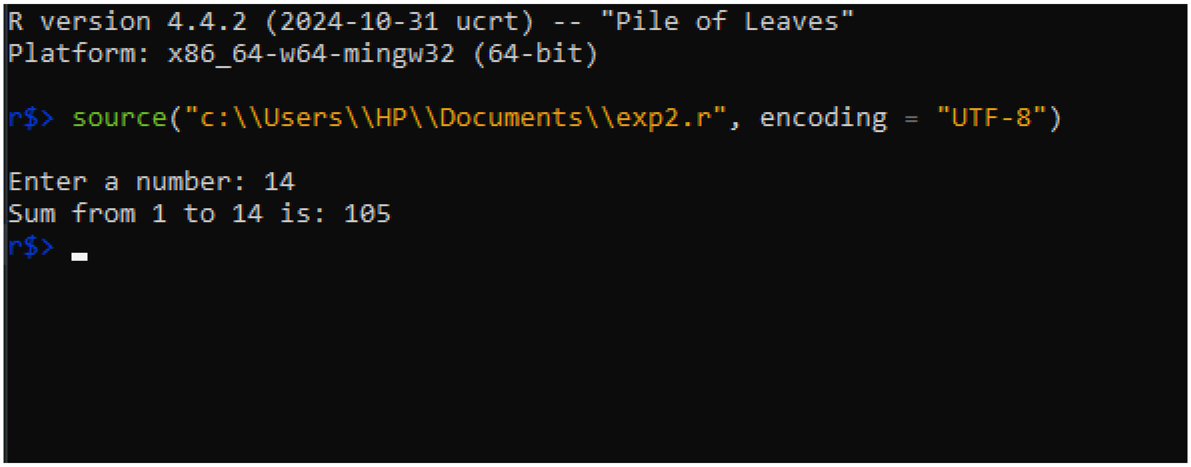
Tools Used: R version 4.4.2

Program:



```
n <- as.integer(readline("Enter a number: "))

cat("Sum from 1 to", n, "is:", sum(1:n), "\n")
```

Output:



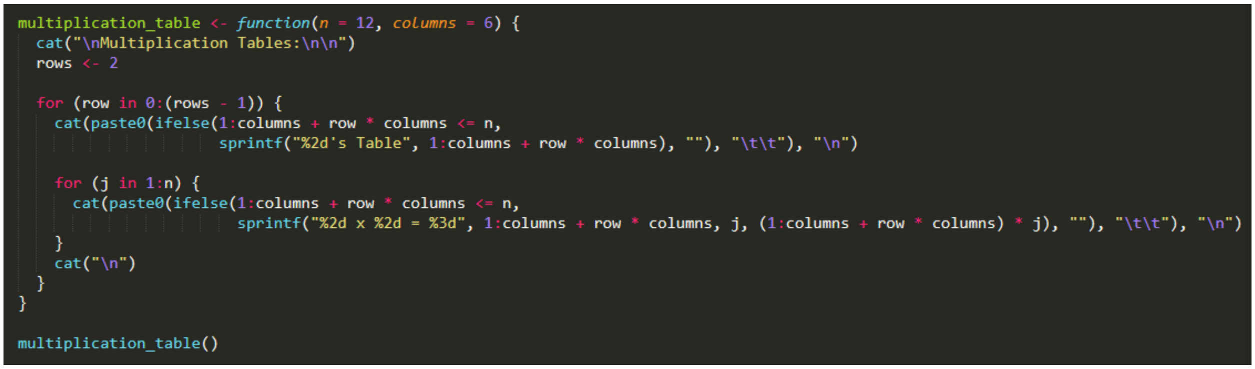
```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("c:\\Users\\HP\\Documents\\exp2.r", encoding = "UTF-8")

Enter a number: 14
Sum from 1 to 14 is: 105
r$> 
```

Experiment – 3

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

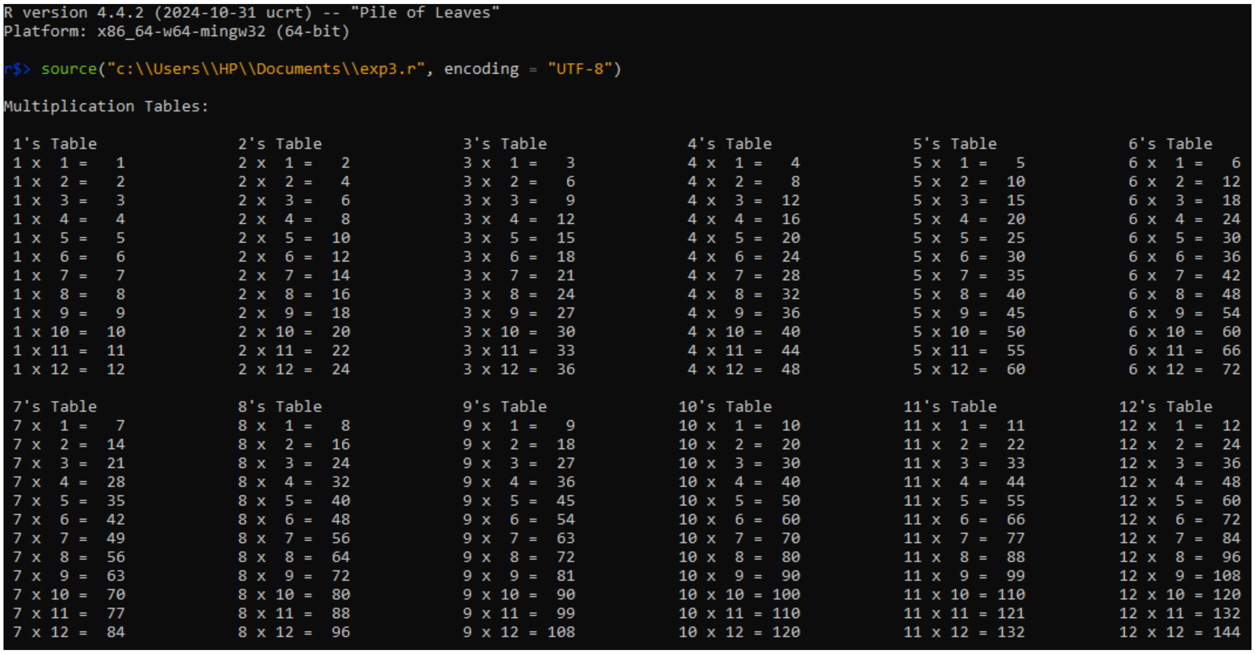
Tools Used: R version 4.4.2

Program:



```r
multiplication_table <- function(n = 12, columns = 6) {
  cat("\nMultiplication Tables:\n\n")
  rows <- 2

  for (row in 0:(rows - 1)) {
    cat(paste0(ifelse(1:columns + row * columns <= n,
                     sprintf("%2d's Table", 1:columns + row * columns), ""), "\t\t"), "\n")

    for (j in 1:n) {
      cat(paste0(ifelse(1:columns + row * columns <= n,
                       sprintf("%2d x %2d = %3d", 1:columns + row * columns, j, (1:columns + row * columns) * j), ""), "\t\t"), "\n")
    }
    cat("\n")
  }
}

multiplication_table()
```

Output:



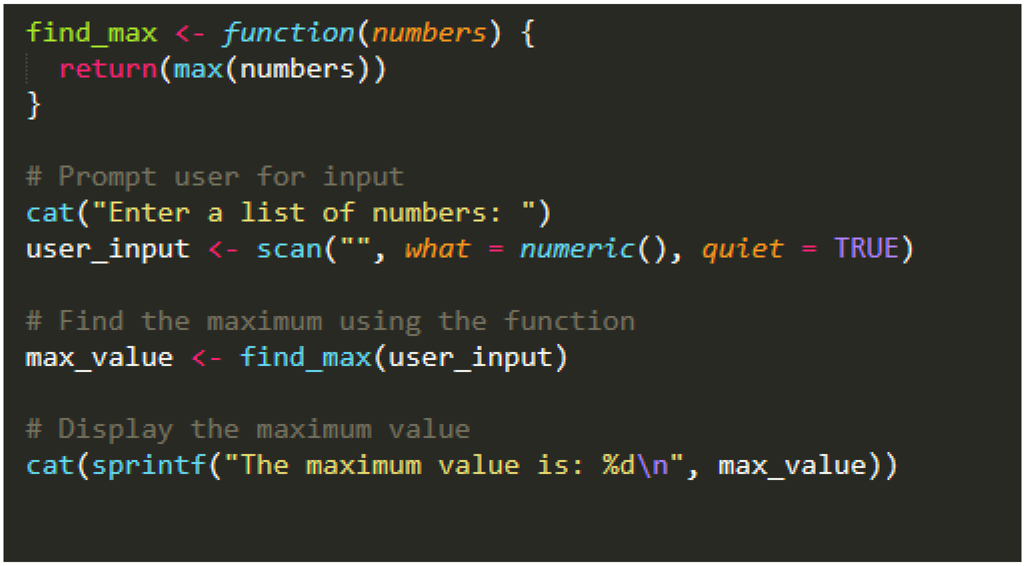
```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("c:\\Users\\HP\\Documents\\exp3.r", encoding = "UTF-8")

Multiplication Tables:

 1's Table               2's Table               3's Table               4's Table               5's Table               6's Table
 1 x  1 =   1            2 x  1 =   2            3 x  1 =   3            4 x  1 =   4            5 x  1 =   5            6 x  1 =   6
 1 x  2 =   2            2 x  2 =   4            3 x  2 =   6            4 x  2 =   8            5 x  2 =  10            6 x  2 =  12
 1 x  3 =   3            2 x  3 =   6            3 x  3 =   9            4 x  3 =  12            5 x  3 =  15            6 x  3 =  18
 1 x  4 =   4            2 x  4 =   8            3 x  4 =  12            4 x  4 =  16            5 x  4 =  20            6 x  4 =  24
 1 x  5 =   5            2 x  5 =  10            3 x  5 =  15            4 x  5 =  20            5 x  5 =  25            6 x  5 =  30
 1 x  6 =   6            2 x  6 =  12            3 x  6 =  18            4 x  6 =  24            5 x  6 =  30            6 x  6 =  36
 1 x  7 =   7            2 x  7 =  14            3 x  7 =  21            4 x  7 =  28            5 x  7 =  35            6 x  7 =  42
 1 x  8 =   8            2 x  8 =  16            3 x  8 =  24            4 x  8 =  32            5 x  8 =  40            6 x  8 =  48
 1 x  9 =   9            2 x  9 =  18            3 x  9 =  27            4 x  9 =  36            5 x  9 =  45            6 x  9 =  54
 1 x 10 =  10            2 x 10 =  20            3 x 10 =  30            4 x 10 =  40            5 x 10 =  50            6 x 10 =  60
 1 x 11 =  11            2 x 11 =  22            3 x 11 =  33            4 x 11 =  44            5 x 11 =  55            6 x 11 =  66
 1 x 12 =  12            2 x 12 =  24            3 x 12 =  36            4 x 12 =  48            5 x 12 =  60            6 x 12 =  72

 7's Table               8's Table               9's Table              10's Table              11's Table              12's Table
 7 x  1 =   7            8 x  1 =   8            9 x  1 =   9           10 x  1 =  10           11 x  1 =  11           12 x  1 =  12
 7 x  2 =  14            8 x  2 =  16            9 x  2 =  18           10 x  2 =  20           11 x  2 =  22           12 x  2 =  24
 7 x  3 =  21            8 x  3 =  24            9 x  3 =  27           10 x  3 =  30           11 x  3 =  33           12 x  3 =  36
 7 x  4 =  28            8 x  4 =  32            9 x  4 =  36           10 x  4 =  40           11 x  4 =  44           12 x  4 =  48
 7 x  5 =  35            8 x  5 =  40            9 x  5 =  45           10 x  5 =  50           11 x  5 =  55           12 x  5 =  60
 7 x  6 =  42            8 x  6 =  48            9 x  6 =  54           10 x  6 =  60           11 x  6 =  66           12 x  6 =  72
 7 x  7 =  49            8 x  7 =  56            9 x  7 =  63           10 x  7 =  70           11 x  7 =  77           12 x  7 =  84
 7 x  8 =  56            8 x  8 =  64            9 x  8 =  72           10 x  8 =  80           11 x  8 =  88           12 x  8 =  96
 7 x  9 =  63            8 x  9 =  72            9 x  9 =  81           10 x  9 =  90           11 x  9 =  99           12 x  9 = 108
 7 x 10 =  70            8 x 10 =  80            9 x 10 =  90           10 x 10 = 100           11 x 10 = 110           12 x 10 = 120
 7 x 11 =  77            8 x 11 =  88            9 x 11 =  99           10 x 11 = 110           11 x 11 = 121           12 x 11 = 132
 7 x 12 =  84            8 x 12 =  96            9 x 12 = 108           10 x 12 = 120           11 x 12 = 132           12 x 12 = 144
```

Experiment – 4

Aim: Write a function that returns the largest element in a list.

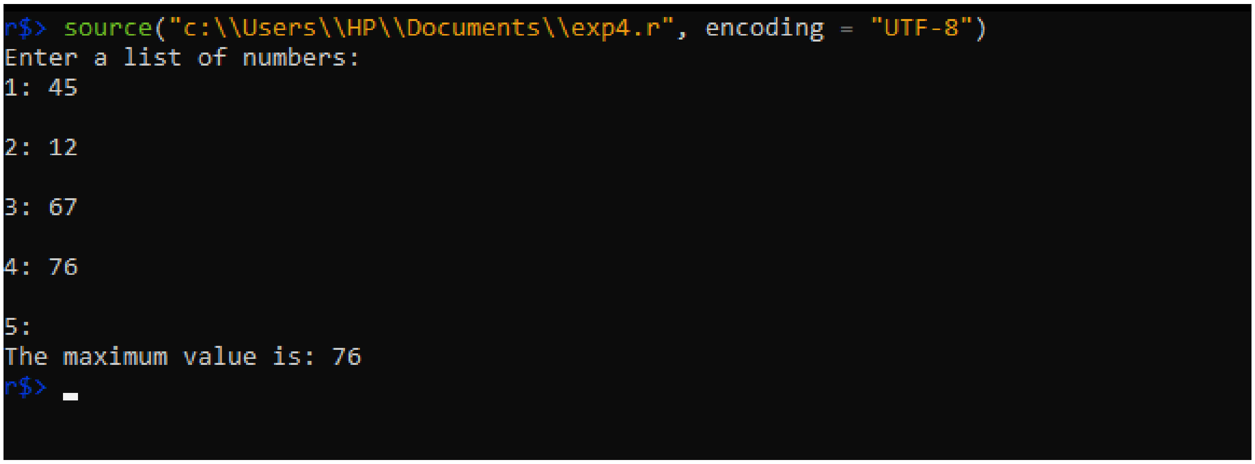
Tools Used: R version 4.4.2

Program:



```r
find_max <- function(numbers) {
  return(max(numbers))
}

# Prompt user for input
cat("Enter a list of numbers: ")
user_input <- scan("", what = numeric(), quiet = TRUE)

# Find the maximum using the function
max_value <- find_max(user_input)

# Display the maximum value
cat(sprintf("The maximum value is: %d\n", max_value))
```

Output:



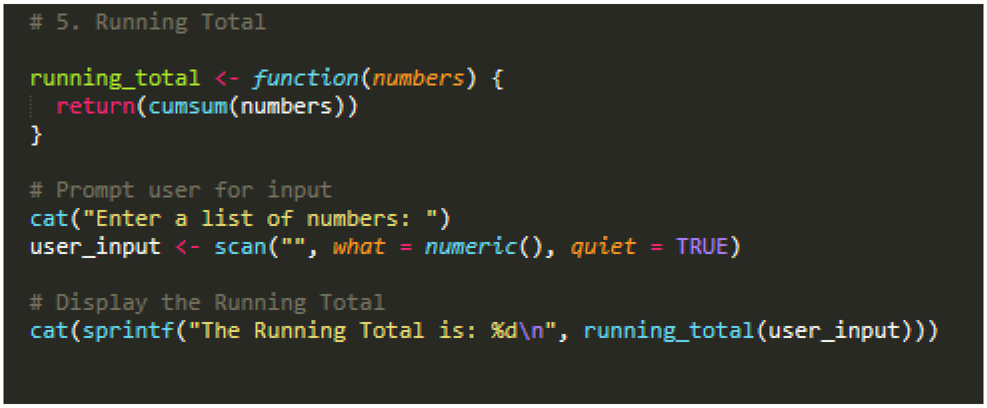
```
r$> source("c:\\Users\\HP\\Documents\\exp4.r", encoding = "UTF-8")
Enter a list of numbers:
1: 45

2: 12

3: 67

4: 76

5:
The maximum value is: 76
r$>
```

Experiment – 5

Aim: Write a function that computes the running total of a list.

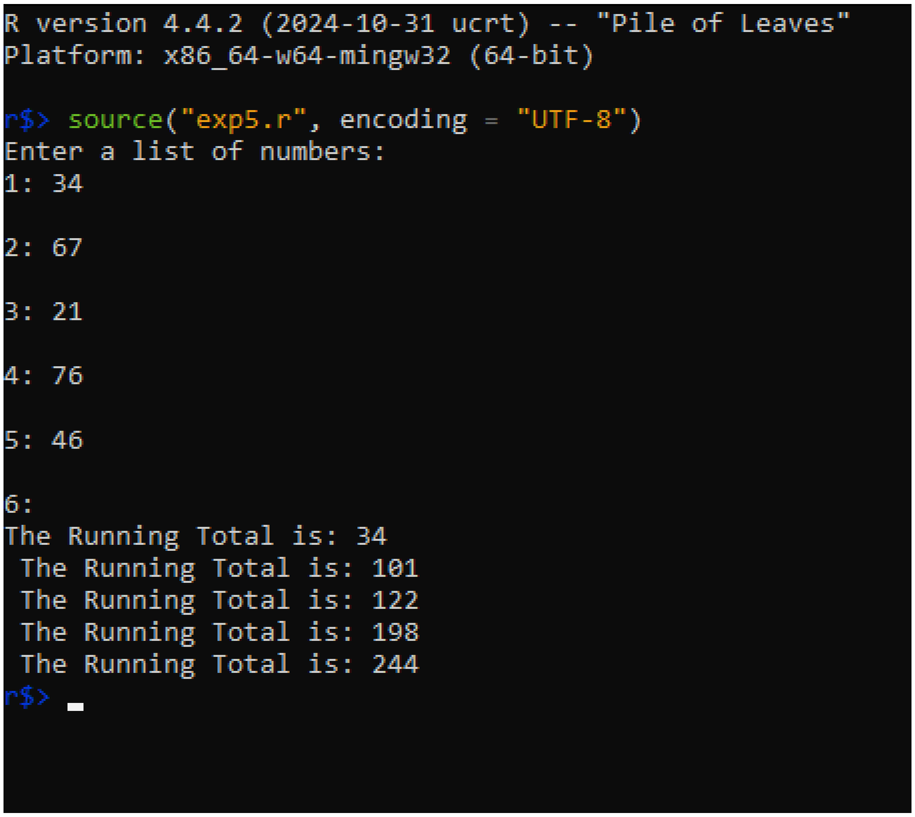
Tools Used: R version 4.4.2

Program:



```r
# 5. Running Total

running_total <- function(numbers) {
  return(cumsum(numbers))
}

# Prompt user for input
cat("Enter a list of numbers: ")
user_input <- scan("", what = numeric(), quiet = TRUE)

# Display the Running Total
cat(sprintf("The Running Total is: %d\n", running_total(user_input)))
```

Output:



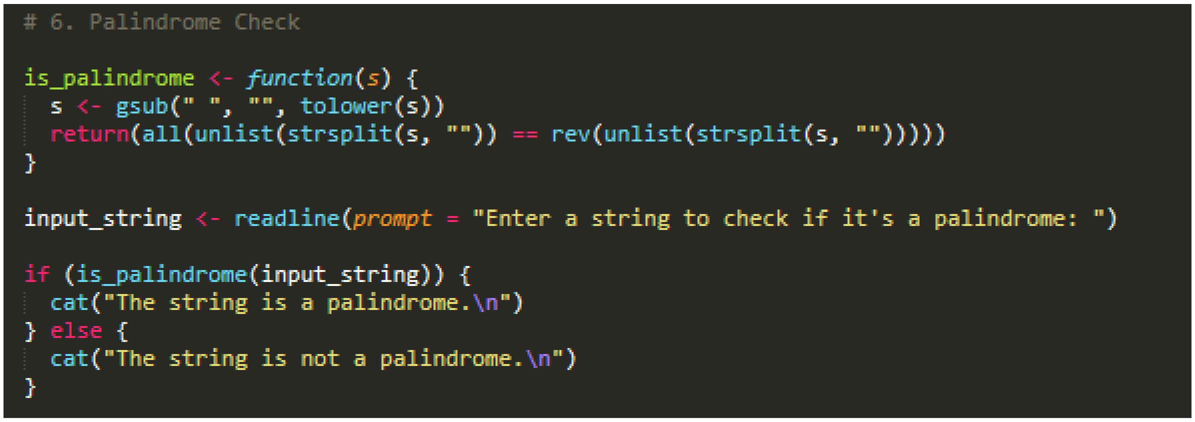
```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("exp5.r", encoding = "UTF-8")
Enter a list of numbers:
1: 34

2: 67

3: 21

4: 76

5: 46

6:
The Running Total is: 34
 The Running Total is: 101
 The Running Total is: 122
 The Running Total is: 198
 The Running Total is: 244
r$> 
```

Experiment – 6-8

Aim: Write a function that tests whether a string is palindrome.

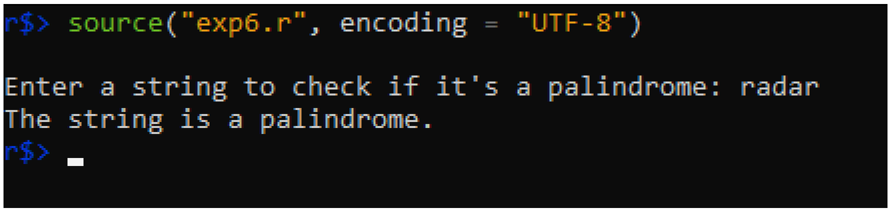
Tools Used: R version 4.4.2

Program:



```r
# 6. Palindrome Check

is_palindrome <- function(s) {
  s <- gsub(" ", "", tolower(s))
  return(all(unlist(strsplit(s, "")) == rev(unlist(strsplit(s, "")))))
}

input_string <- readline(prompt = "Enter a string to check if it's a palindrome: ")

if (is_palindrome(input_string)) {
  cat("The string is a palindrome.\n")
} else {
  cat("The string is not a palindrome.\n")
}
```

Output:



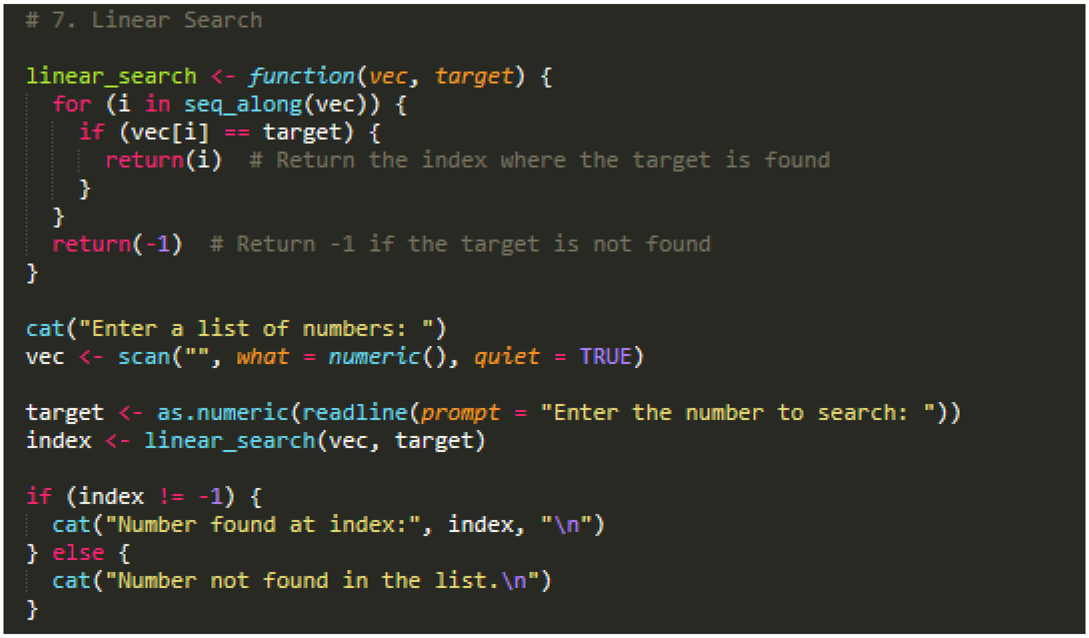
```
r$> source("exp6.r", encoding = "UTF-8")

Enter a string to check if it's a palindrome: radar
The string is a palindrome.
r$> 
```

Experiment – 9-11

Aim: Implement linear search.

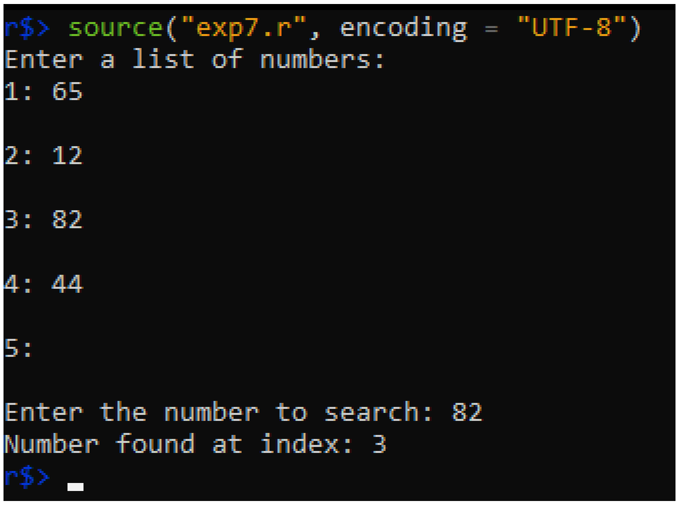
Tools Used: R version 4.4.2

Program:



```r
# 7. Linear Search

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

cat("Enter a list of numbers: ")
vec <- scan("", what = numeric(), quiet = TRUE)

target <- as.numeric(readline(prompt = "Enter the number to search: "))
index <- linear_search(vec, target)

if (index != -1) {
  cat("Number found at index:", index, "\n")
} else {
  cat("Number not found in the list.\n")
}
```

Output:



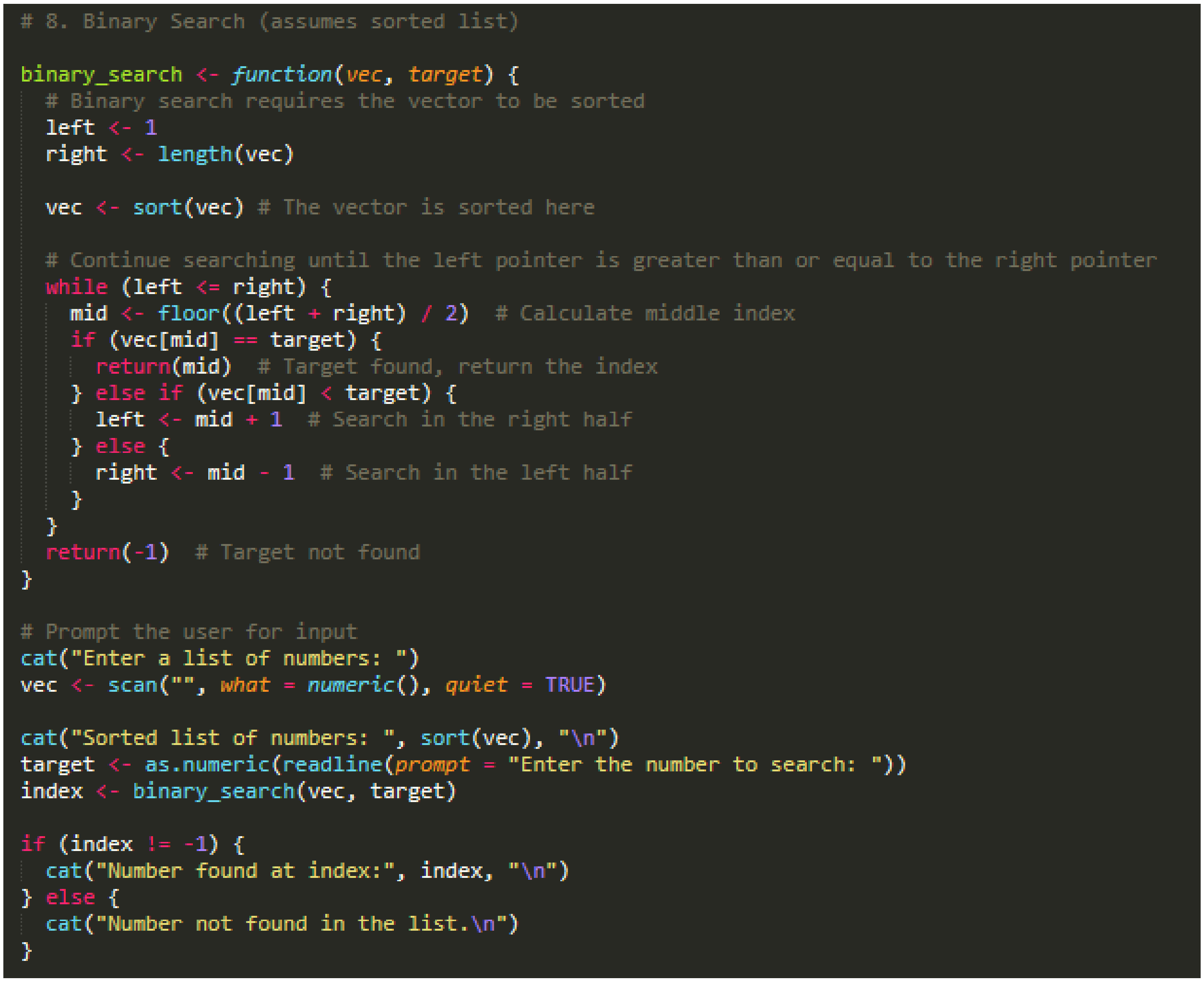
```
r$> source("exp7.r", encoding = "UTF-8")
Enter a list of numbers:
1: 65

2: 12

3: 82

4: 44

5:

Enter the number to search: 82
Number found at index: 3
r$>
```

Experiment – 12-14

Aim: Implement binary search.

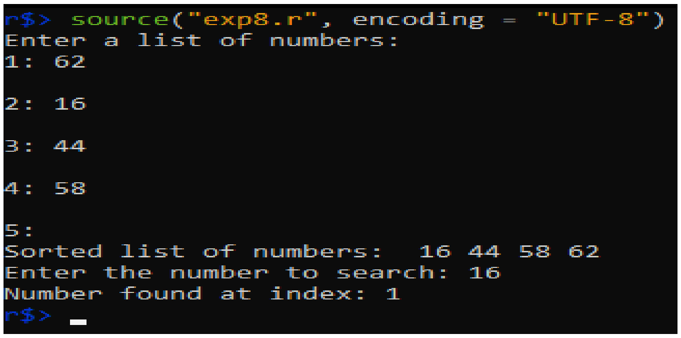
Tools Used: R version 4.4.2

Program:



```r
# 8. Binary Search (assumes sorted list)

binary_search <- function(vec, target) {
  # Binary search requires the vector to be sorted
  left <- 1
  right <- length(vec)

  vec <- sort(vec) # The vector is sorted here

  # Continue searching until the left pointer is greater than or equal to the right pointer
  while (left <= right) {
    mid <- floor((left + right) / 2)  # Calculate middle index
    if (vec[mid] == target) {
      return(mid)  # Target found, return the index
    } else if (vec[mid] < target) {
      left <- mid + 1  # Search in the right half
    } else {
      right <- mid - 1  # Search in the left half
    }
  }
  return(-1)  # Target not found
}

# Prompt the user for input
cat("Enter a list of numbers: ")
vec <- scan("", what = numeric(), quiet = TRUE)

cat("Sorted list of numbers: ", sort(vec), "\n")
target <- as.numeric(readline(prompt = "Enter the number to search: "))
index <- binary_search(vec, target)

if (index != -1) {
  cat("Number found at index:", index, "\n")
} else {
  cat("Number not found in the list.\n")
}
```

Output:



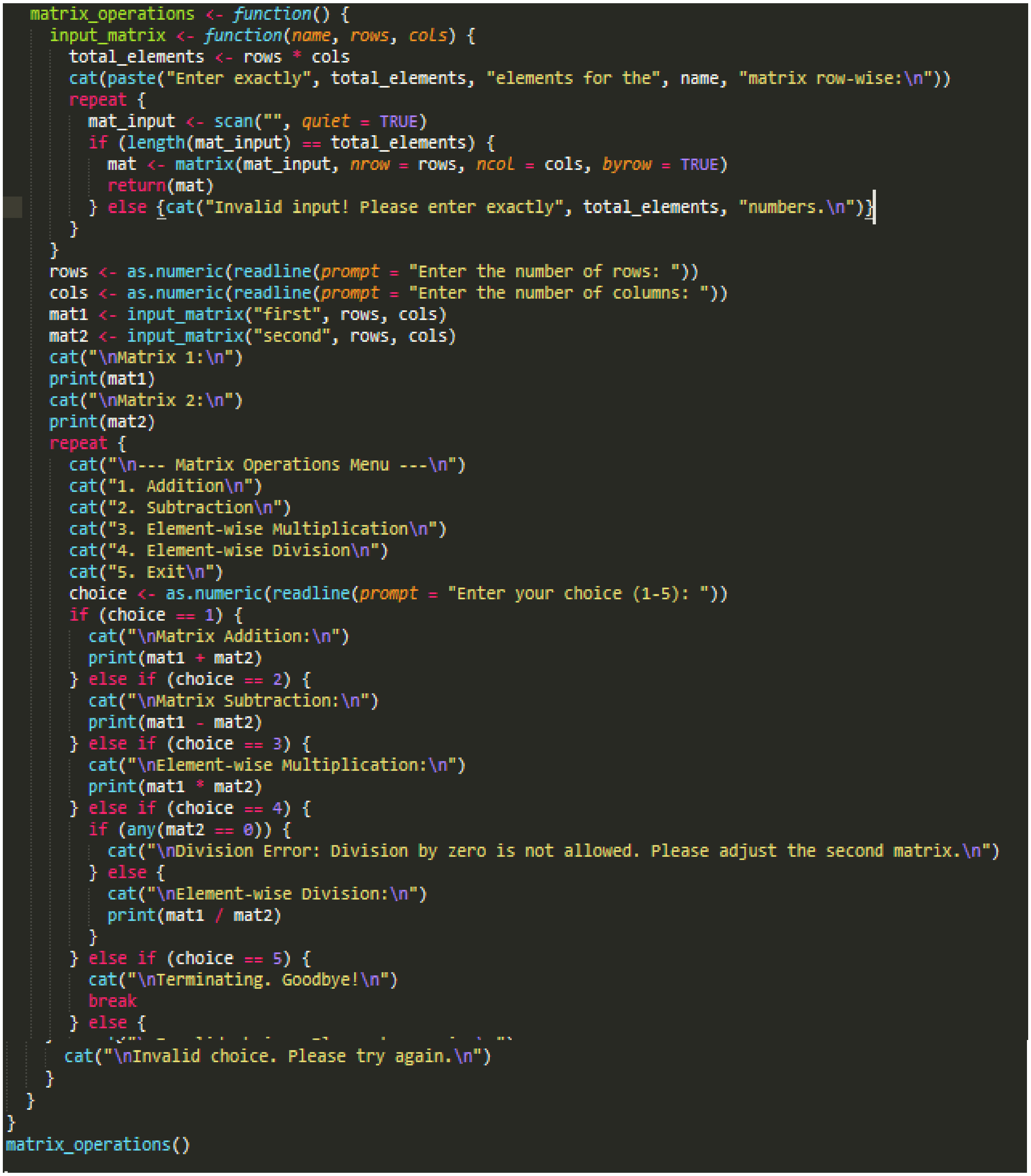
```
r$> source("exp8.r", encoding = "UTF-8")
Enter a list of numbers:
1: 62

2: 16

3: 44

4: 58

5:
Sorted list of numbers:  16 44 58 62
Enter the number to search: 16
Number found at index: 1
r$> _
```

Experiment – 15-17

Aim: Implement matrices addition, subtraction, multiplication and division.

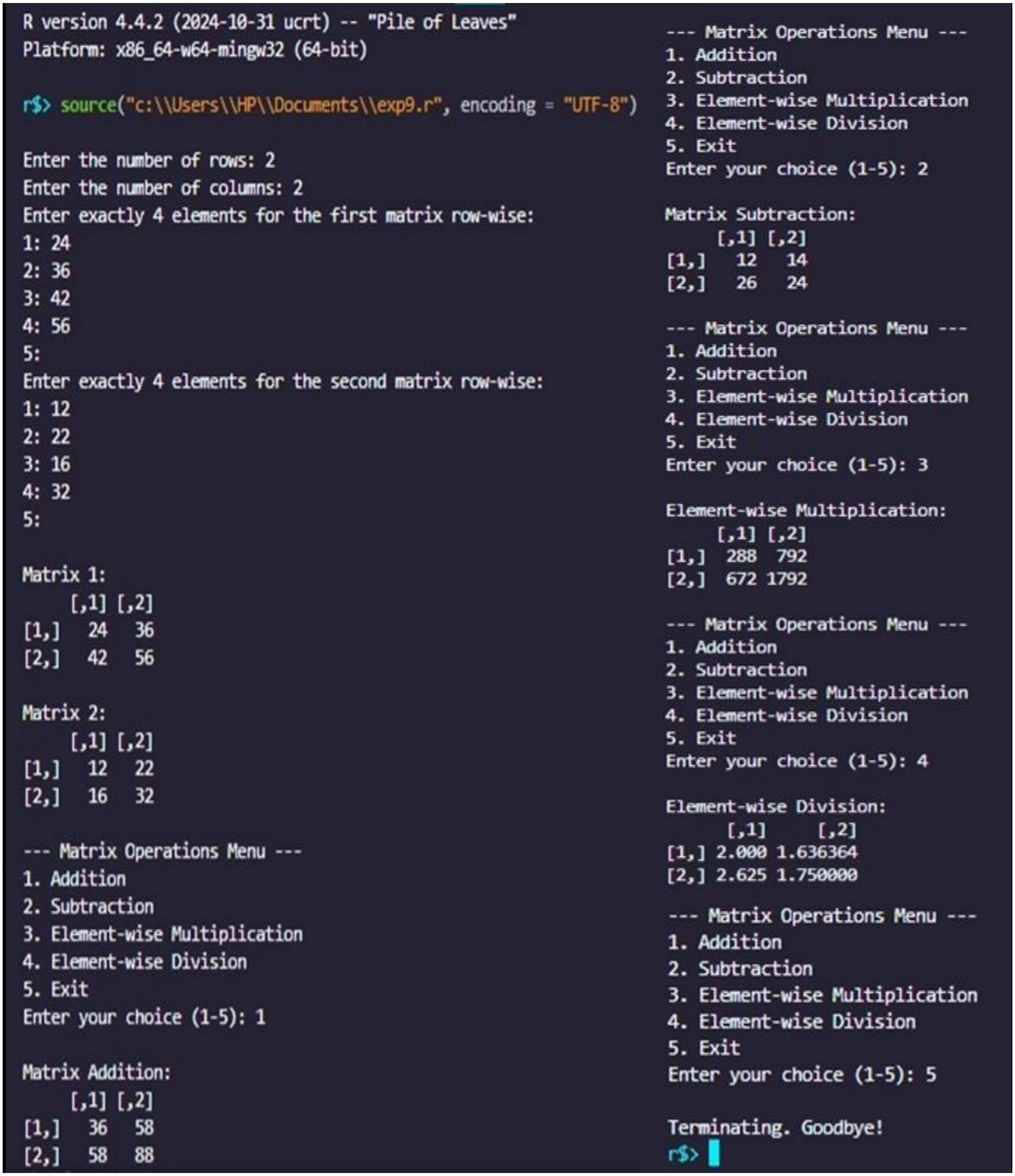
Tools Used: R version 4.4.2

Program:



```
matrix_operations <- function() {
  input_matrix <- function(name, rows, cols) {
    total_elements <- rows * cols
    cat(paste("Enter exactly", total_elements, "elements for the", name, "matrix row-wise:\n"))
    repeat {
      mat_input <- scan("", quiet = TRUE)
      if (length(mat_input) == total_elements) {
        mat <- matrix(mat_input, nrow = rows, ncol = cols, byrow = TRUE)
        return(mat)
      } else {cat("Invalid input! Please enter exactly", total_elements, "numbers.\n")}
    }
  }
  rows <- as.numeric(readline(prompt = "Enter the number of rows: "))
  cols <- as.numeric(readline(prompt = "Enter the number of columns: "))
  mat1 <- input_matrix("first", rows, cols)
  mat2 <- input_matrix("second", rows, cols)
  cat("\nMatrix 1:\n")
  print(mat1)
  cat("\nMatrix 2:\n")
  print(mat2)
  repeat {
    cat("\n--- Matrix Operations Menu ---\n")
    cat("1. Addition\n")
    cat("2. Subtraction\n")
    cat("3. Element-wise Multiplication\n")
    cat("4. Element-wise Division\n")
    cat("5. Exit\n")
    choice <- as.numeric(readline(prompt = "Enter your choice (1-5): "))
    if (choice == 1) {
      cat("\nMatrix Addition:\n")
      print(mat1 + mat2)
    } else if (choice == 2) {
      cat("\nMatrix Subtraction:\n")
      print(mat1 - mat2)
    } else if (choice == 3) {
      cat("\nElement-wise Multiplication:\n")
      print(mat1 * mat2)
    } else if (choice == 4) {
      if (any(mat2 == 0)) {
        cat("\nDivision Error: Division by zero is not allowed. Please adjust the second matrix.\n")
      } else {
        cat("\nElement-wise Division:\n")
        print(mat1 / mat2)
      }
    } else if (choice == 5) {
      cat("\nTerminating. Goodbye!\n")
      break
    } else {
      cat("\nInvalid choice. Please try again.\n")
    }
  }
}
matrix_operations()
```

Output:



```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("c:\\Users\\HP\\Documents\\exp9.r", encoding = "UTF-8")

Enter the number of rows: 2
Enter the number of columns: 2
Enter exactly 4 elements for the first matrix row-wise:
1: 24
2: 36
3: 42
4: 56
5:
Enter exactly 4 elements for the second matrix row-wise:
1: 12
2: 22
3: 16
4: 32
5:

Matrix 1:
     [,1] [,2]
[1,]   24   36
[2,]   42   56

Matrix 2:
     [,1] [,2]
[1,]   12   22
[2,]   16   32

--- Matrix Operations Menu ---
1. Addition
2. Subtraction
3. Element-wise Multiplication
4. Element-wise Division
5. Exit
Enter your choice (1-5): 1

Matrix Addition:
     [,1] [,2]
[1,]   36   58
[2,]   58   88
```

```
--- Matrix Operations Menu ---
1. Addition
2. Subtraction
3. Element-wise Multiplication
4. Element-wise Division
5. Exit
Enter your choice (1-5): 2

Matrix Subtraction:
     [,1] [,2]
[1,]   12   14
[2,]   26   24

--- Matrix Operations Menu ---
1. Addition
2. Subtraction
3. Element-wise Multiplication
4. Element-wise Division
5. Exit
Enter your choice (1-5): 3

Element-wise Multiplication:
     [,1] [,2]
[1,]  288  792
[2,]  672 1792

--- Matrix Operations Menu ---
1. Addition
2. Subtraction
3. Element-wise Multiplication
4. Element-wise Division
5. Exit
Enter your choice (1-5): 4

Element-wise Division:
      [,1]     [,2]
[1,] 2.000 1.636364
[2,] 2.625 1.750000

--- Matrix Operations Menu ---
1. Addition
2. Subtraction
3. Element-wise Multiplication
4. Element-wise Division
5. Exit
Enter your choice (1-5): 5

Terminating. Goodbye!
r$>
```

Experiment – 18-20

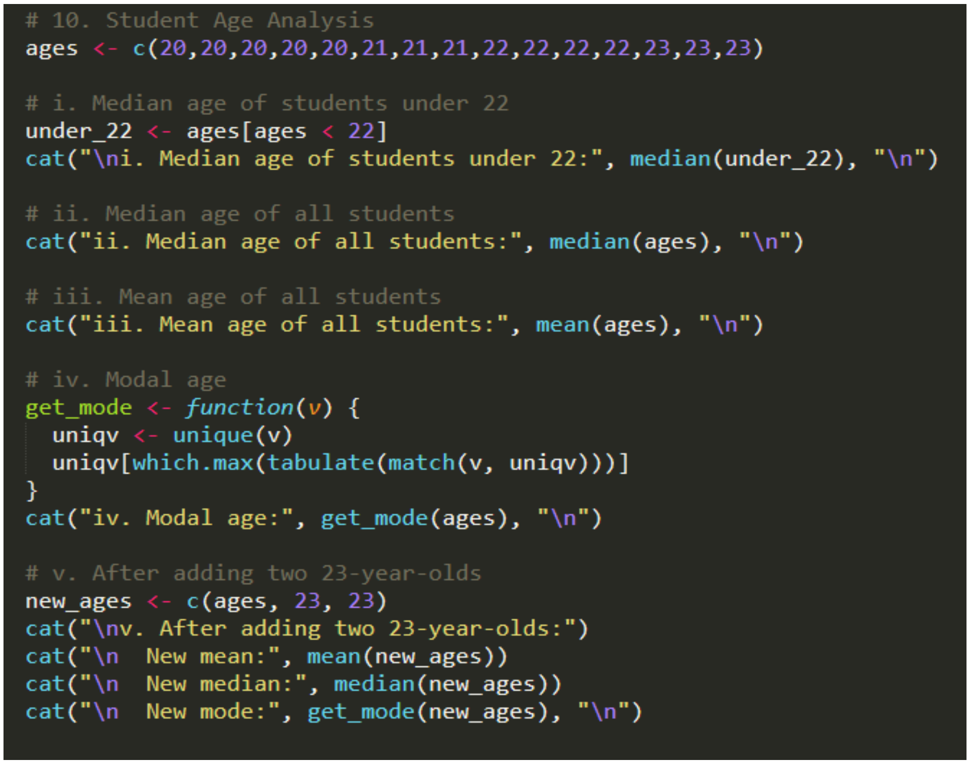
Aim: Fifteen students were enrolled in a course. There ages were:

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 of all students**
v. **Two more students enter the class. The age of both students is 23. What is now mean, mode and median?**

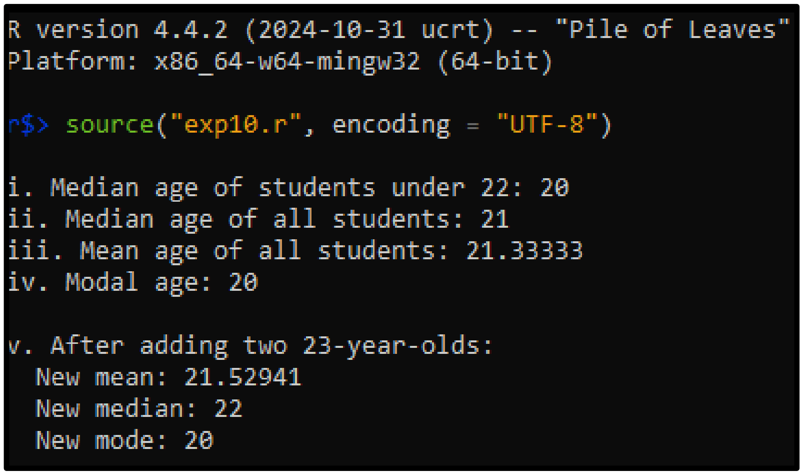
Tools Used: R version 4.4.2

Program:



```r
# 10. Student Age Analysis
ages <- c(20,20,20,20,20,21,21,21,22,22,22,22,23,23,23)

# i. Median age of students under 22
under_22 <- ages[ages < 22]
cat("\ni. Median age of students under 22:", median(under_22), "\n")

# ii. Median age of all students
cat("ii. Median age of all students:", median(ages), "\n")

# iii. Mean age of all students
cat("iii. Mean age of all students:", mean(ages), "\n")

# iv. Modal age
get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
cat("iv. Modal age:", get_mode(ages), "\n")

# v. After adding two 23-year-olds
new_ages <- c(ages, 23, 23)
cat("\nv. After adding two 23-year-olds:")
cat("\n  New mean:", mean(new_ages))
cat("\n  New median:", median(new_ages))
cat("\n  New mode:", get_mode(new_ages), "\n")
```

Output:



```
R version 4.4.2 (2024-10-31 ucrt) -- "Pile of Leaves"
Platform: x86_64-w64-mingw32 (64-bit)

r$> source("exp10.r", encoding = "UTF-8")

i. Median age of students under 22: 20
ii. Median age of all students: 21
iii. Mean age of all students: 21.33333
iv. Modal age: 20

v. After adding two 23-year-olds:
  New mean: 21.52941
  New median: 22
  New mode: 20
```