

*** R Programming Language ***

INDEX PAGE

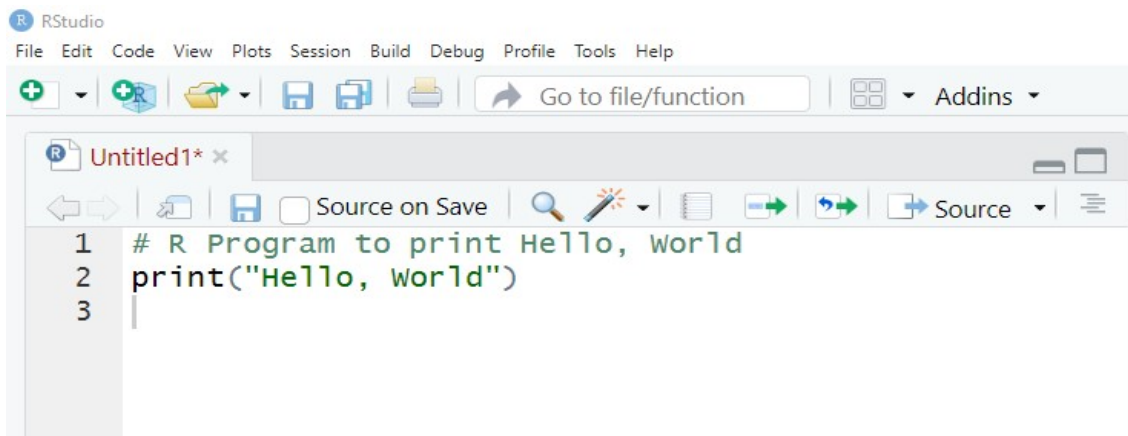
S.NO	Questions	Page No.
01.	Write a program print "Hello world" to the screen.	2
02.	Write a program that asks the user for a number n and prints the sum of the 1 to n.	3
03.	Write a program that prints a multiplication table for numbers up to 12.	4
04.	Write a function that returns the largest element in a list.	5
05.	Write a function that computes the running total of a list.	6
06.	Write a function that tests whether a string is Palindrome.	7
07.	Implement the following sorting algorithms: Selection sort, Insertion sort, Bubble sort.	8-10
08.	Implement linear search.	11
09.	Implement binary search.	12
10.	Implement matrices addition, subtraction and Multiplication.	13-14

01.=> Print "Hello World" to the Screen.

> Here a simple R Program that prints "Hello, World":

Syntax=> print("Hello, World")

Code >

A screenshot of the RStudio IDE. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, printing, and navigating. The main editor window shows a script titled 'Untitled1*' with the following code:

```
1 # R Program to print Hello, World
2 print("Hello, World")
3
```

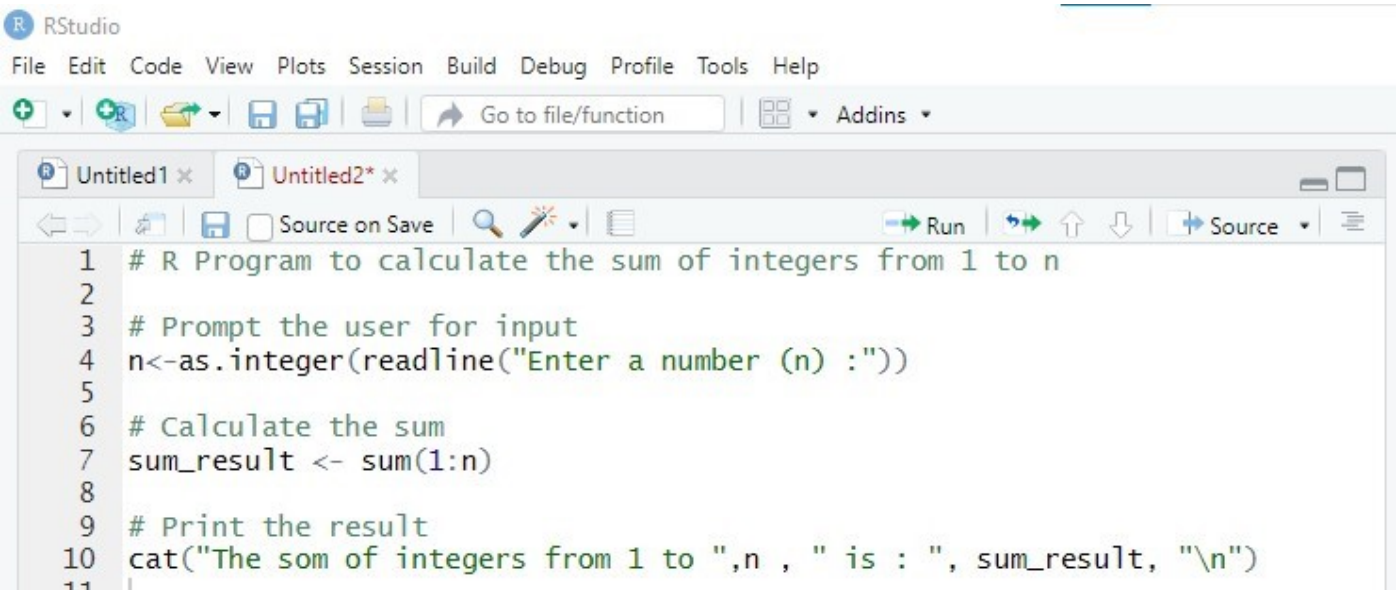
Output >

```
> # R Program to print Hello, World
> print("Hello, World")
[1] "Hello, World"
>
```

O2. > Write a program that asks the user for a number n and prints the sum of the 1 to n.

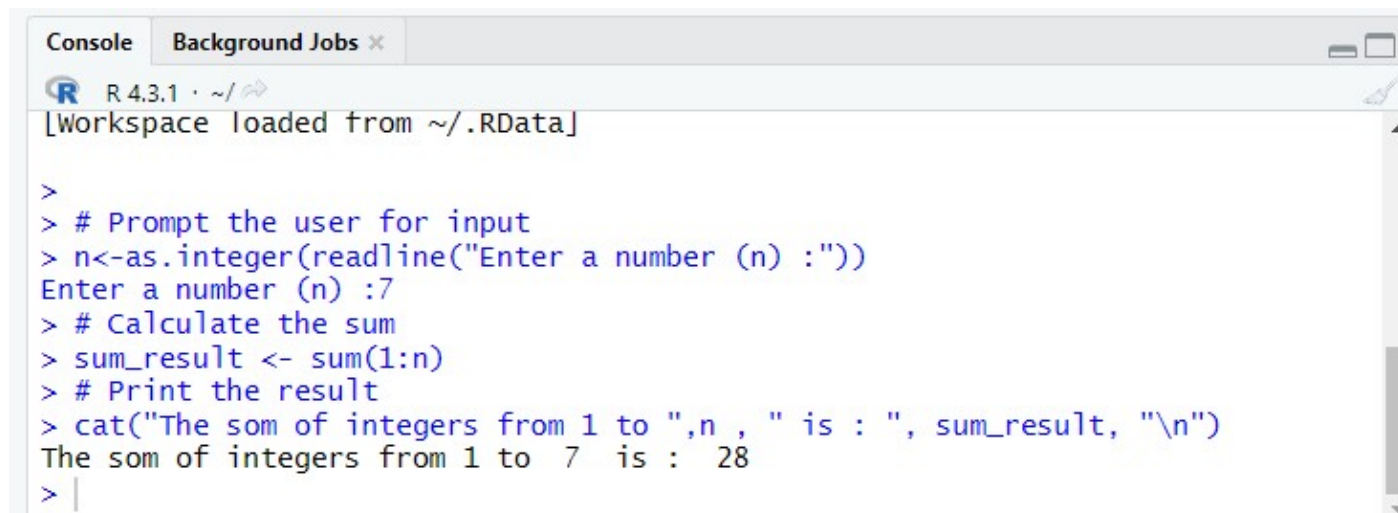
>that prompts the user for a number n and prints the sum of integers from 1 to n;

Code >



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Untitled1 x Untitled2* x
Source on Save Run
1 # R Program to calculate the sum of integers from 1 to n
2
3 # Prompt the user for input
4 n<-as.integer(readline("Enter a number (n) :"))
5
6 # Calculate the sum
7 sum_result <- sum(1:n)
8
9 # Print the result
10 cat("The som of integers from 1 to ",n , " is : ", sum_result, "\n")
11
```

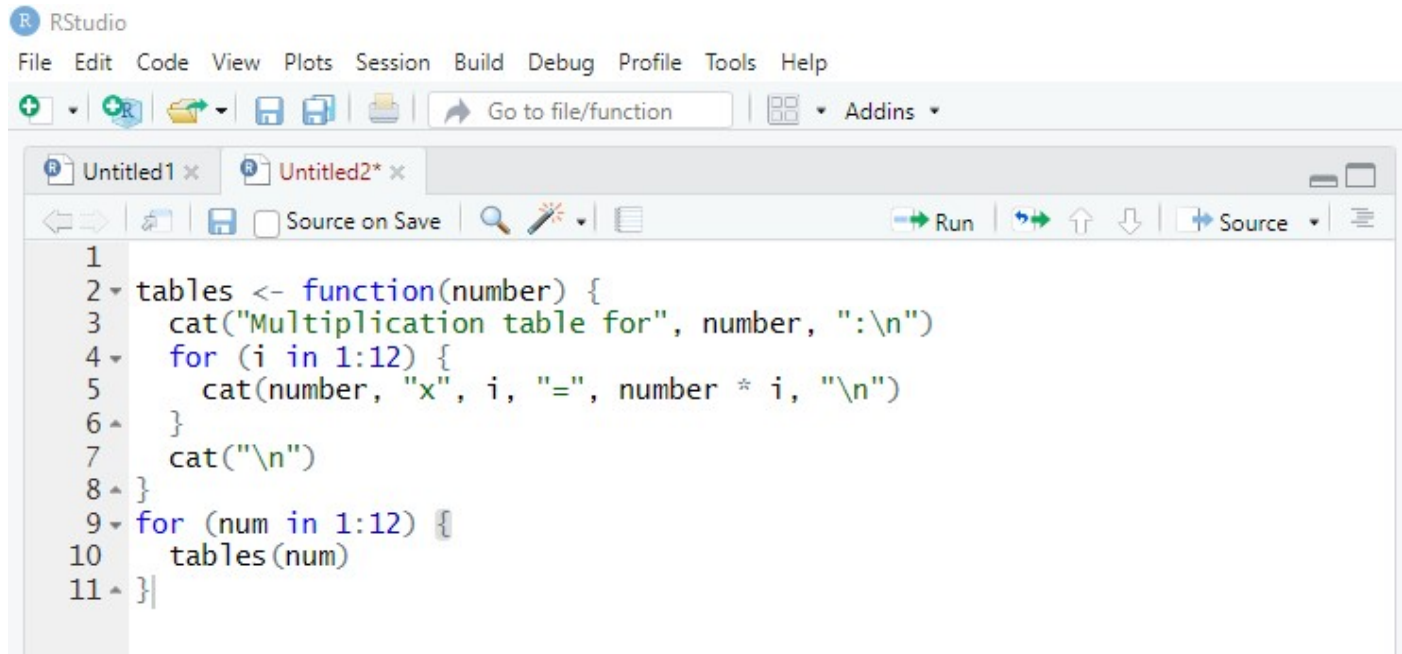
Output >



```
Console Background Jobs x
R 4.3.1 ~/
[Workspace loaded from ~/.RData]
>
> # Prompt the user for input
> n<-as.integer(readline("Enter a number (n) :"))
Enter a number (n) :7
> # Calculate the sum
> sum_result <- sum(1:n)
> # Print the result
> cat("The som of integers from 1 to ",n , " is : ", sum_result, "\n")
The som of integers from 1 to 7 is : 28
> |
```

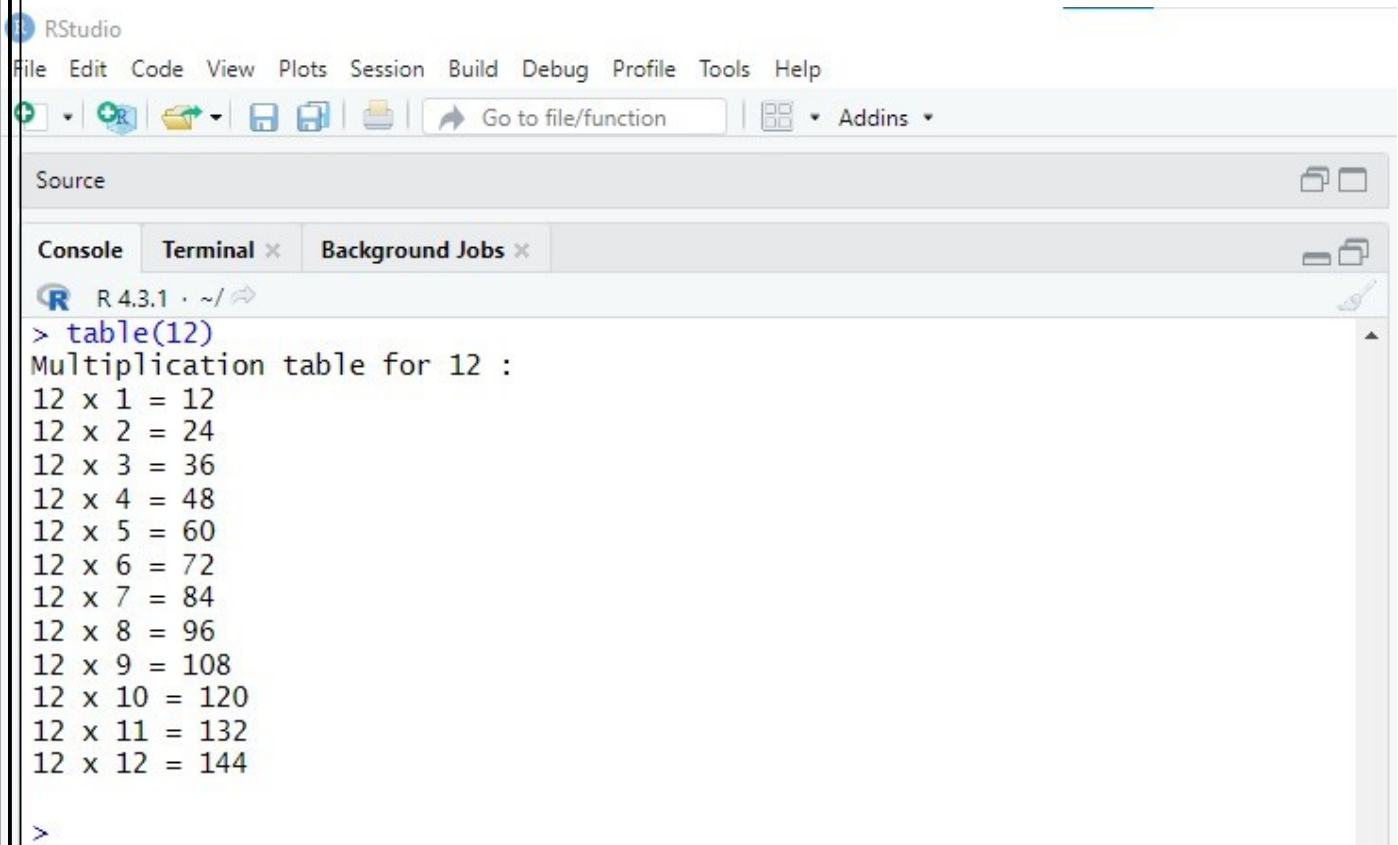
03. > Write a program that prints a multiplication table for numbers up to 12.

Code >



```
1
2 tables <- function(number) {
3   cat("Multiplication table for", number, ":\n")
4   for (i in 1:12) {
5     cat(number, "x", i, "=", number * i, "\n")
6   }
7   cat("\n")
8 }
9 for (num in 1:12) {
10  tables(num)
11 }
```

Output >

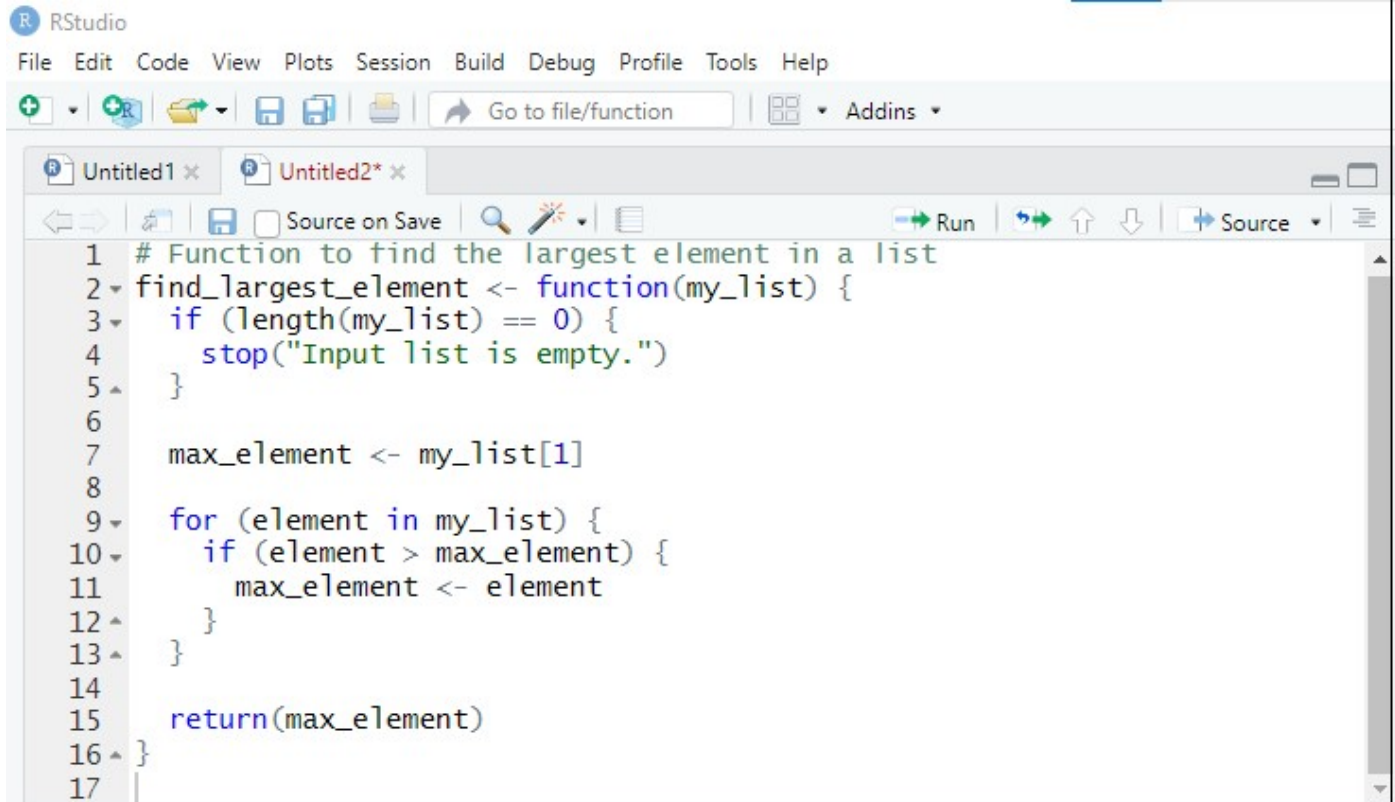


```
R 4.3.1 ~ /
> table(12)
Multiplication table for 12 :
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132
12 x 12 = 144

>
```

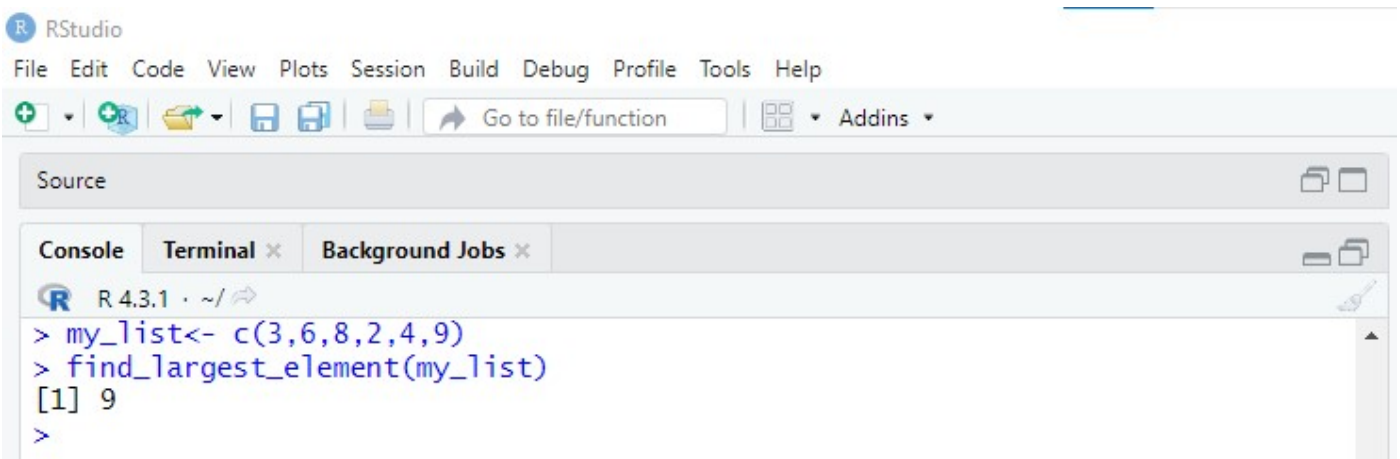
04. > Write a function that returns the largest element in a list.

Code >



```
1 # Function to find the largest element in a list
2 find_largest_element <- function(my_list) {
3   if (length(my_list) == 0) {
4     stop("Input list is empty.")
5   }
6
7   max_element <- my_list[1]
8
9   for (element in my_list) {
10    if (element > max_element) {
11      max_element <- element
12    }
13  }
14
15  return(max_element)
16 }
17
```

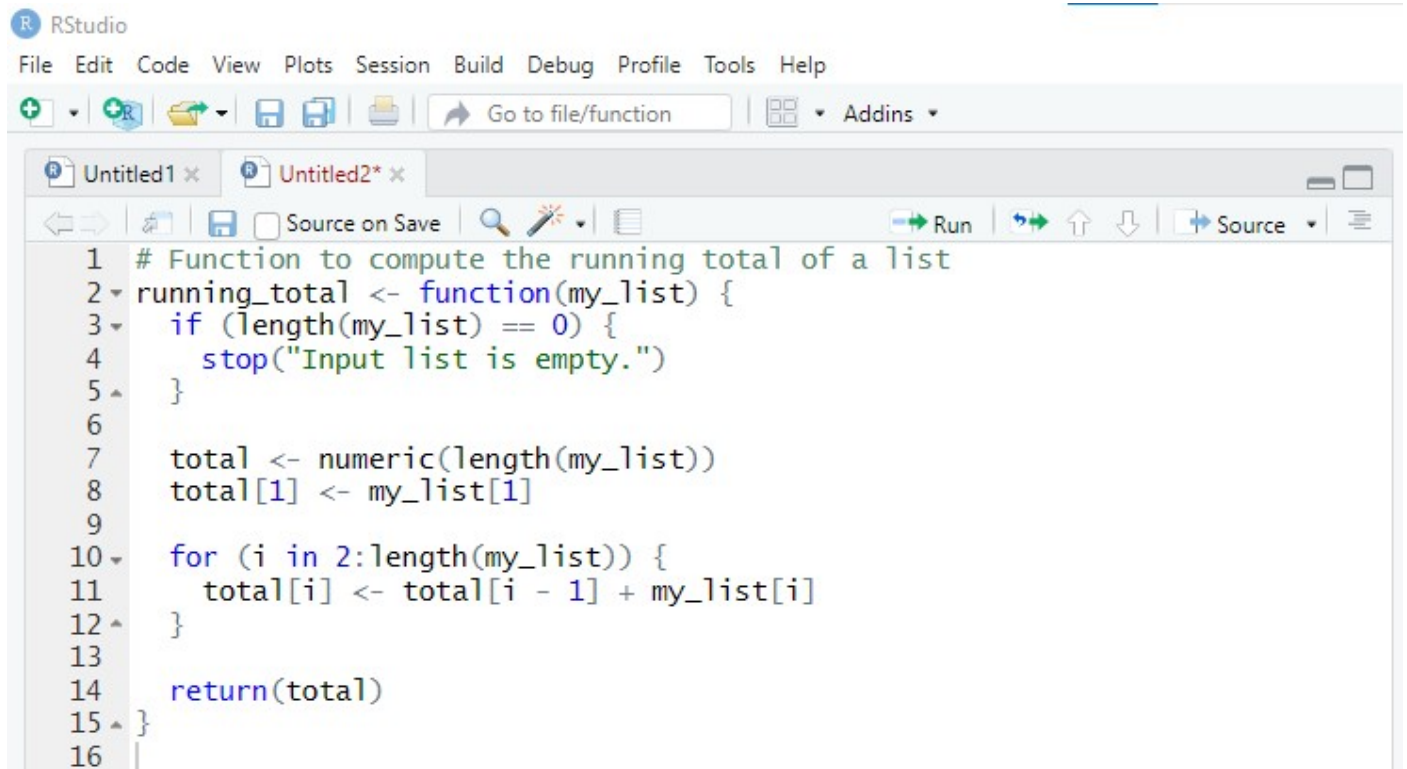
Output >



```
R 4.3.1 ~/> my_list<- c(3,6,8,2,4,9)
> find_largest_element(my_list)
[1] 9
>
```

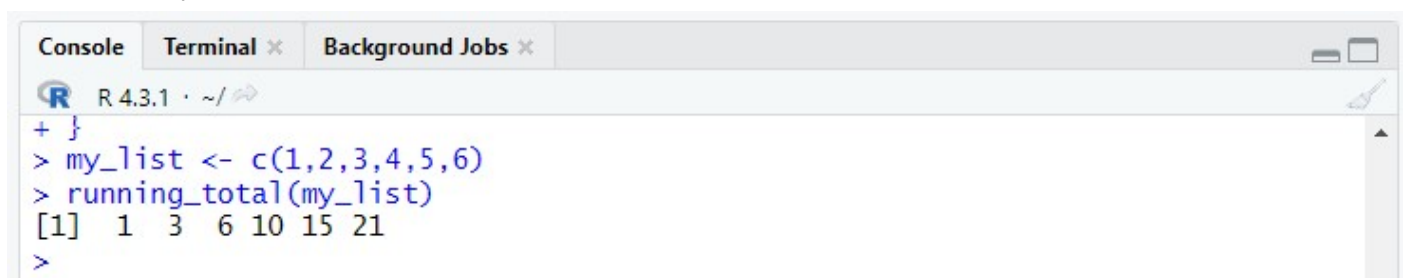
05. > Write a function that computes the running total of a list.

Code >



```
1 # Function to compute the running total of a list
2 running_total <- function(my_list) {
3   if (length(my_list) == 0) {
4     stop("Input list is empty.")
5   }
6
7   total <- numeric(length(my_list))
8   total[1] <- my_list[1]
9
10  for (i in 2:length(my_list)) {
11    total[i] <- total[i - 1] + my_list[i]
12  }
13
14  return(total)
15 }
16
```

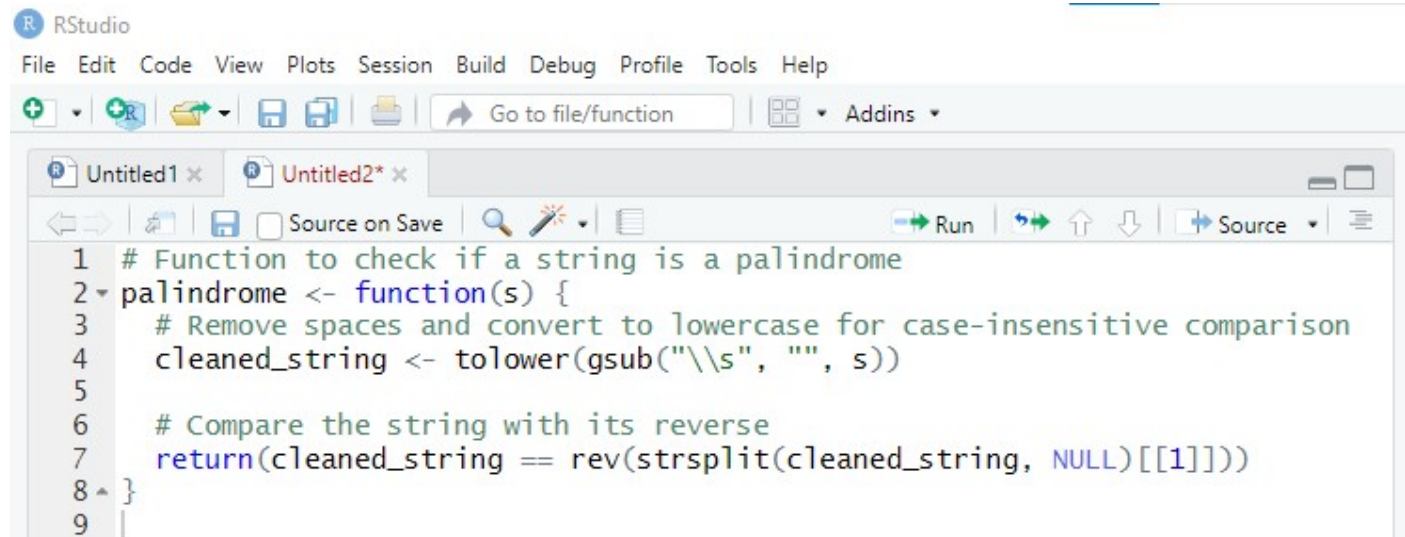
Output >



```
R 4.3.1 ~/>
+ }
> my_list <- c(1,2,3,4,5,6)
> running_total(my_list)
[1] 1 3 6 10 15 21
>
```

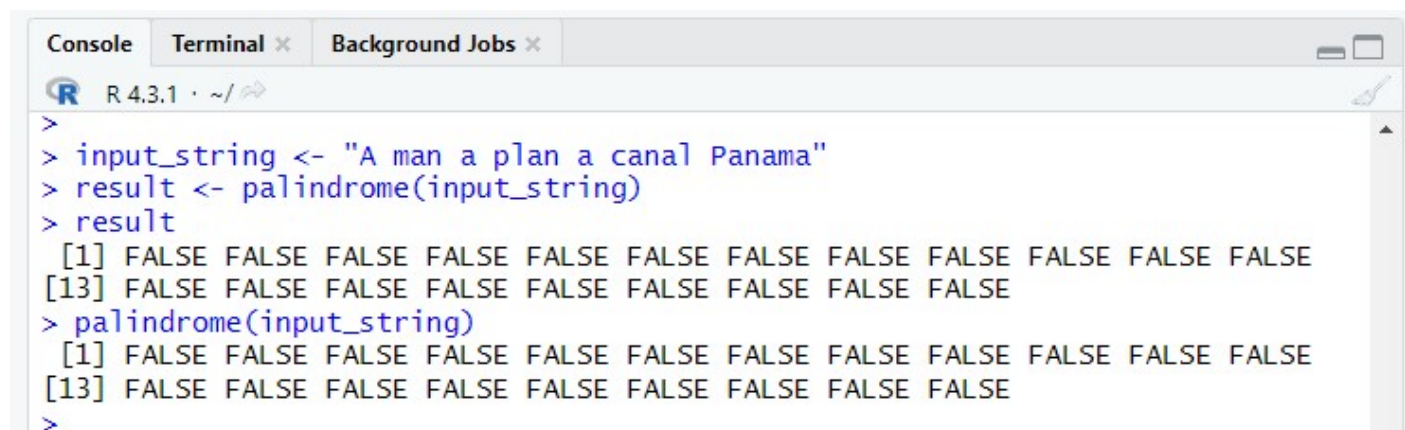

06. > Write a function that tests whether a string is Palindrome.

Code >



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ [R] [Folder] [Save] [Source on Save] [Go to file/function] [Addins]
[Untitled1] [Untitled2*]
[Run] [Source]
1 # Function to check if a string is a palindrome
2 palindrome <- function(s) {
3   # Remove spaces and convert to lowercase for case-insensitive comparison
4   cleaned_string <- tolower(gsub("\\s", "", s))
5
6   # Compare the string with its reverse
7   return(cleaned_string == rev(strsplit(cleaned_string, NULL)[[1]]))
8 }
9
```

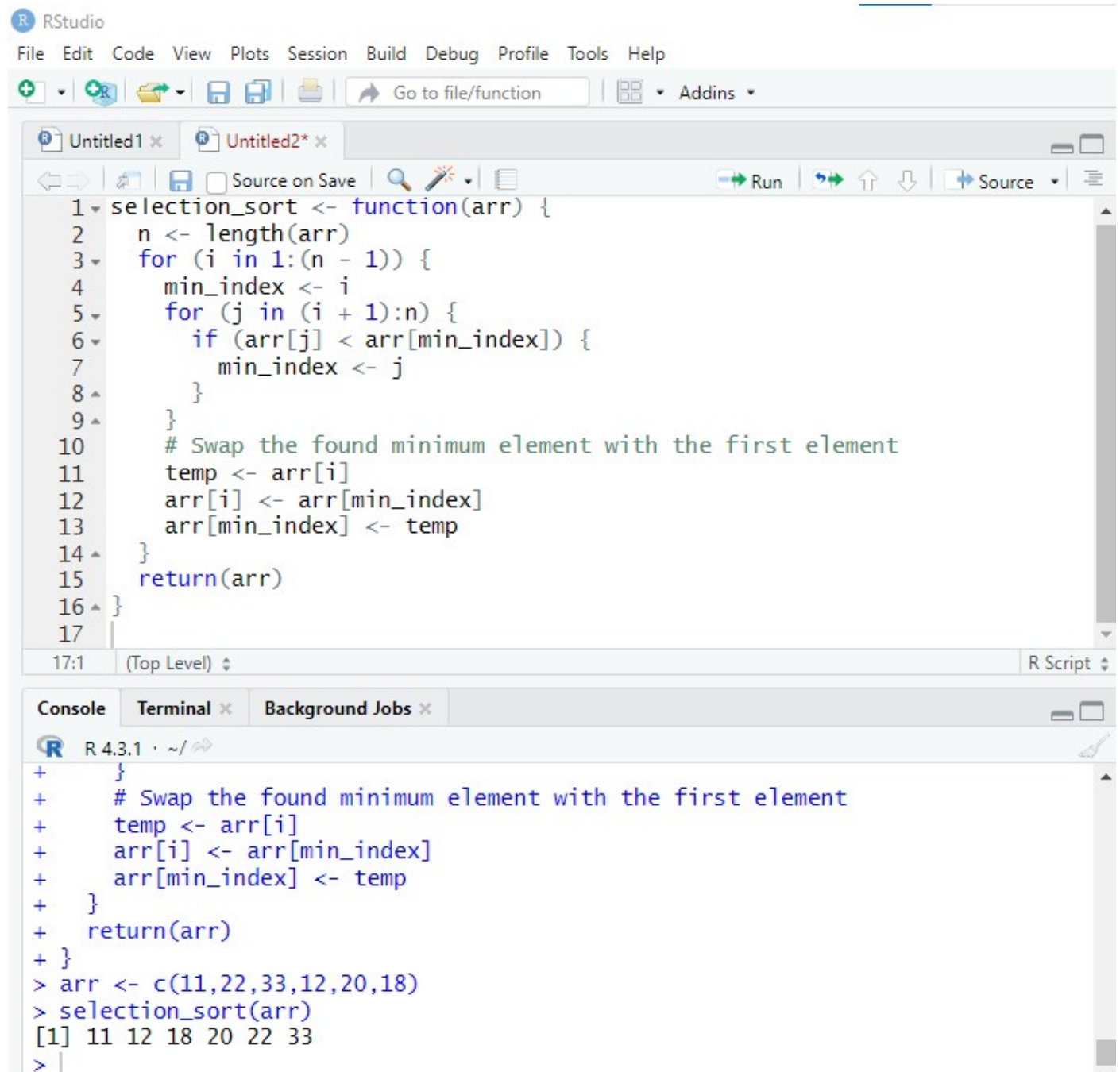
Output >



```
Console Terminal Background Jobs
R 4.3.1 ~
>
> input_string <- "A man a plan a canal Panama"
> result <- palindrome(input_string)
> result
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> palindrome(input_string)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
```

07. > Implement the following sorting algorithms: Selection sort, Insertion sort, Bubble sort.

Code > Selection Sort



```
1 selection_sort <- function(arr) {
2   n <- length(arr)
3   for (i in 1:(n - 1)) {
4     min_index <- i
5     for (j in (i + 1):n) {
6       if (arr[j] < arr[min_index]) {
7         min_index <- j
8       }
9     }
10    # Swap the found minimum element with the first element
11    temp <- arr[i]
12    arr[i] <- arr[min_index]
13    arr[min_index] <- temp
14  }
15  return(arr)
16 }
17
```

```
+ }
+ # Swap the found minimum element with the first element
+ temp <- arr[i]
+ arr[i] <- arr[min_index]
+ arr[min_index] <- temp
+ }
+ return(arr)
+ }
> arr <- c(11,22,33,12,20,18)
> selection_sort(arr)
[1] 11 12 18 20 22 33
>
```


Code > Insertion Sort

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

```
1 insertion_sort <- function(arr) {  
2   n <- length(arr)  
3   for (i in 2:n) {  
4     key <- arr[i]  
5     j <- i - 1  
6     while (j > 0 && arr[j] > key) {  
7       arr[j + 1] <- arr[j]  
8       j <- j - 1  
9     }  
10    arr[j + 1] <- key  
11  }  
12  return(arr)  
13 }  
14 |
```

14:1 (Top Level)

R Script

Console Terminal x Background Jobs x

```
R 4.3.1 ~/  
+ j <- i - 1  
+ while (j > 0 && arr[j] > key) {  
+   arr[j + 1] <- arr[j]  
+   j <- j - 1  
+ }  
+ arr[j + 1] <- key  
+ }  
+ return(arr)  
+ }  
> insertion_sort(arr)  
[1] 11 12 18 20 22 33  
> |
```

Code > Bubble Sort

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

```
1 bubble_sort <- function(arr) {  
2   n <- length(arr)  
3   for (i in 1:(n - 1)) {  
4     for (j in 1:(n - i)) {  
5       if (arr[j] > arr[j + 1]) {  
6         # Swap if the element found is greater than the next element  
7         temp <- arr[j]  
8         arr[j] <- arr[j + 1]  
9         arr[j + 1] <- temp  
10      }  
11    }  
12  }  
13  return(arr)  
14 }  
15
```

15:1 (Top Level)

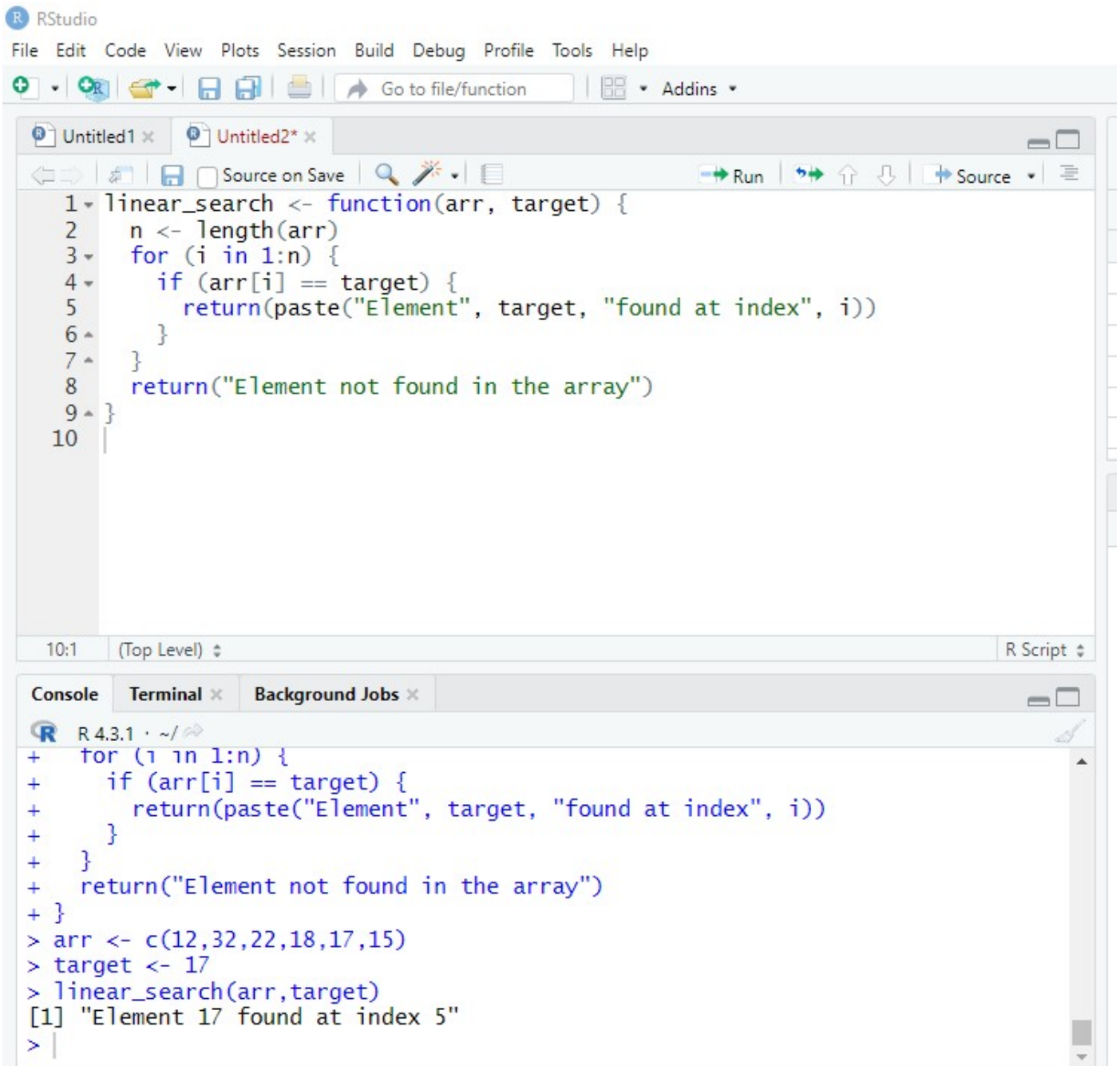
R Script

Console Terminal x Background Jobs x

```
R 4.3.1 ~/  
+ # Swap if the element found is greater than the next element  
+ temp <- arr[j]  
+ arr[j] <- arr[j + 1]  
+ arr[j + 1] <- temp  
+ }  
+ }  
+ }  
+ return(arr)  
+ }  
> bubble_sort(arr)  
[1] 11 12 18 20 22 33  
>
```

08. > Implement linear search.

Code >



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main editor window displays a script named 'Untitled2*' with the following R code:

```
1 linear_search <- function(arr, target) {  
2   n <- length(arr)  
3   for (i in 1:n) {  
4     if (arr[i] == target) {  
5       return(paste("Element", target, "found at index", i))  
6     }  
7   }  
8   return("Element not found in the array")  
9 }  
10
```

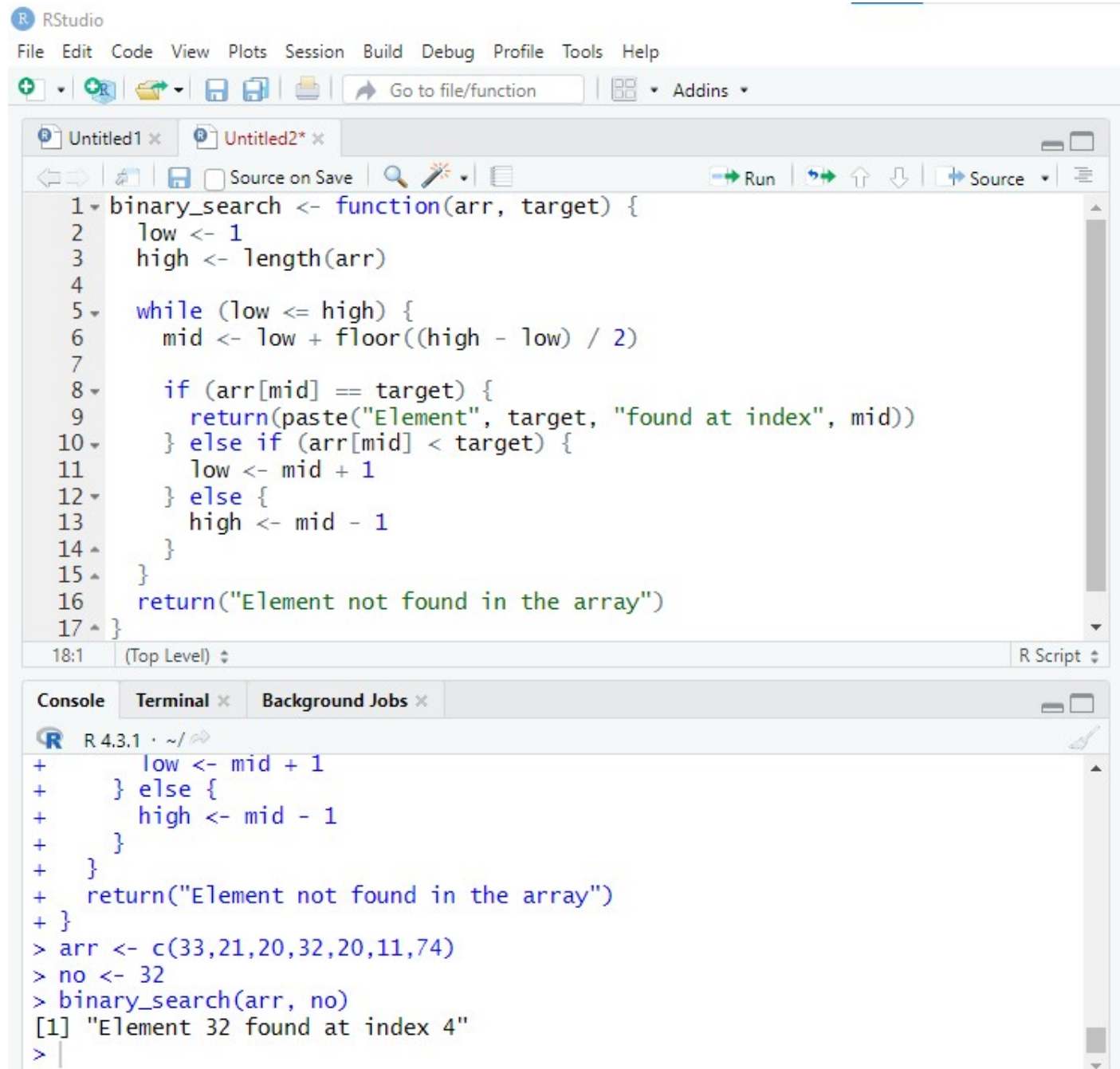
The status bar at the bottom of the editor shows '10:1 (Top Level)' and 'R Script'.

Below the editor is the Console window, which shows the execution of the code:

```
R 4.3.1 ~/  
+ for (i in 1:n) {  
+   if (arr[i] == target) {  
+     return(paste("Element", target, "found at index", i))  
+   }  
+ }  
+ return("Element not found in the array")  
+ }  
> arr <- c(12,32,22,18,17,15)  
> target <- 17  
> linear_search(arr,target)  
[1] "Element 17 found at index 5"  
>
```

09. > Implement binary search.

Code >



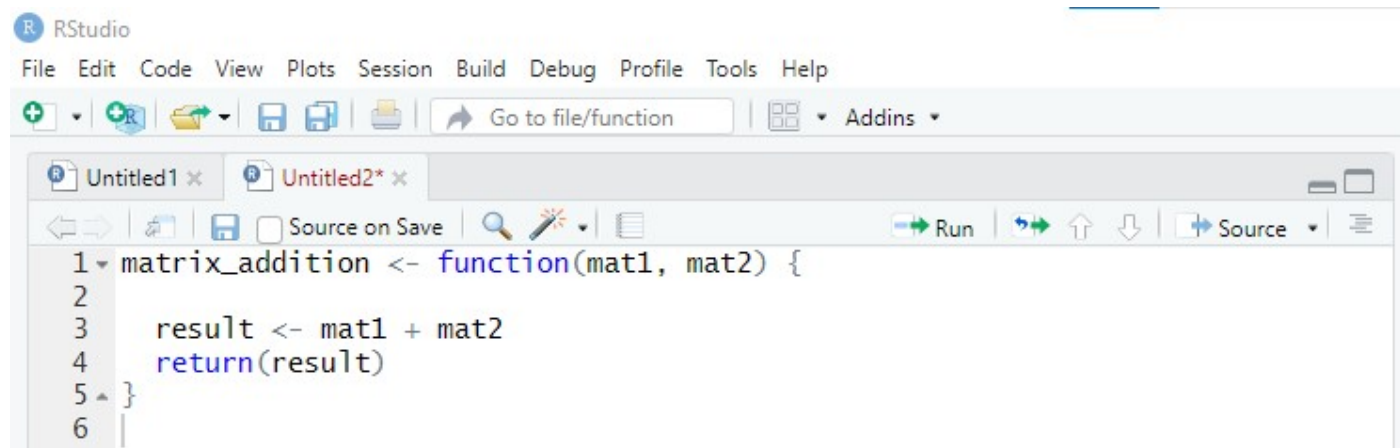
The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main editor window displays a function named `binary_search` in a script file. The function takes an array `arr` and a `target` as input. It uses a while loop to find the target element by repeatedly halving the search range. If the target is found, it returns a message with the index; otherwise, it returns a message stating the element was not found. The console window at the bottom shows the execution of the function with a sample array and target, resulting in the output: `[1] "Element 32 found at index 4"`.

```
1 binary_search <- function(arr, target) {  
2   low <- 1  
3   high <- length(arr)  
4  
5   while (low <= high) {  
6     mid <- low + floor((high - low) / 2)  
7  
8     if (arr[mid] == target) {  
9       return(paste("Element", target, "found at index", mid))  
10    } else if (arr[mid] < target) {  
11      low <- mid + 1  
12    } else {  
13      high <- mid - 1  
14    }  
15  }  
16  return("Element not found in the array")  
17 }
```

```
+   low <- mid + 1  
+ } else {  
+   high <- mid - 1  
+ }  
+ }  
+ return("Element not found in the array")  
+ }  
> arr <- c(33,21,20,32,20,11,74)  
> no <- 32  
> binary_search(arr, no)  
[1] "Element 32 found at index 4"  
>
```

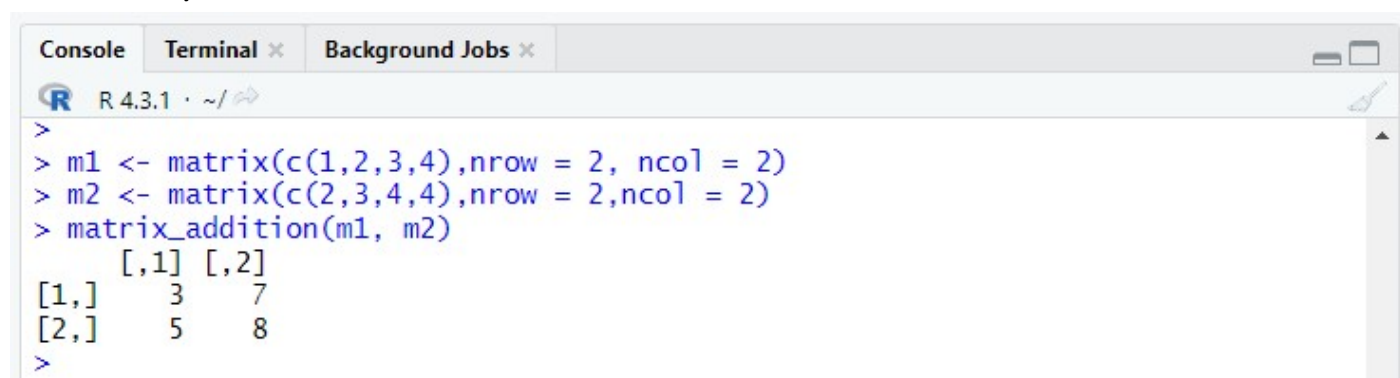
10. > Implement matrices addition, subtraction and Multiplication.

Code > Addition matrices



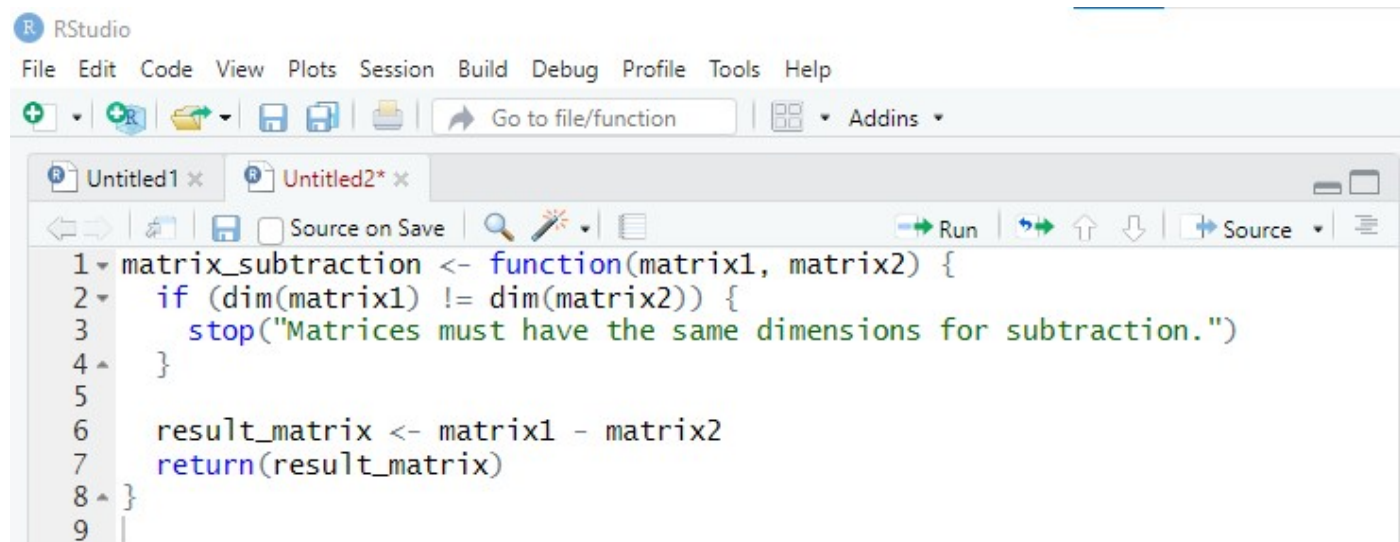
```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ [R] [Save] [Run] [Source] [Go to file/function] [Addins]
[Untitled1] [Untitled2*]
[Run] [Source]
1 matrix_addition <- function(mat1, mat2) {
2
3   result <- mat1 + mat2
4   return(result)
5 }
6
```

Output >



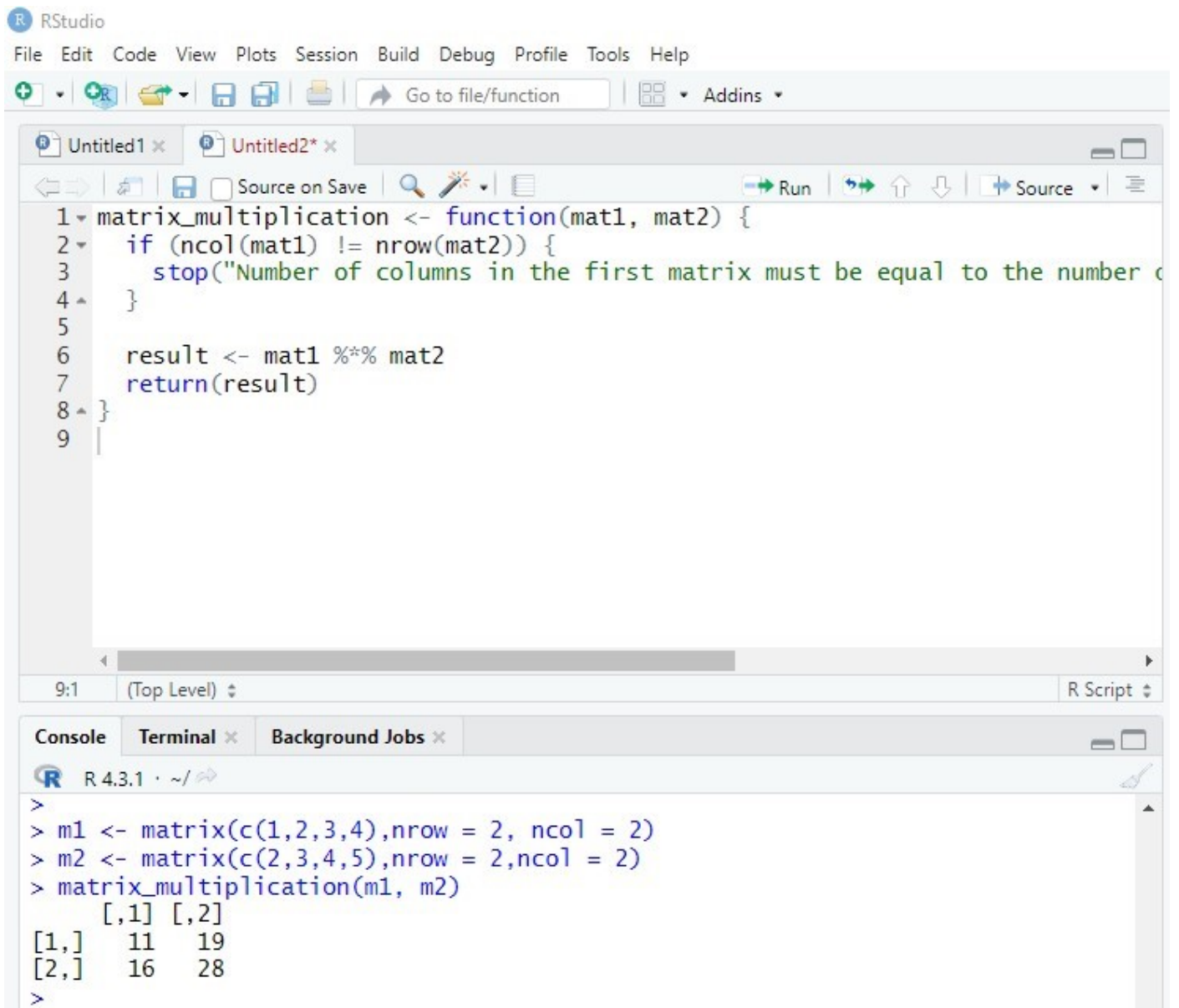
```
R 4.3.1 ~
>
> m1 <- matrix(c(1,2,3,4),nrow = 2, ncol = 2)
> m2 <- matrix(c(2,3,4,4),nrow = 2,ncol = 2)
> matrix_addition(m1, m2)
      [,1] [,2]
[1,]    3    7
[2,]    5    8
>
```

Code > Matrix Subtraction



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ [R] [Save] [Run] [Source] [Go to file/function] [Addins]
[Untitled1] [Untitled2*]
[Run] [Source]
1 matrix_subtraction <- function(matrix1, matrix2) {
2   if (dim(matrix1) != dim(matrix2)) {
3     stop("Matrices must have the same dimensions for subtraction.")
4   }
5
6   result_matrix <- matrix1 - matrix2
7   return(result_matrix)
8 }
9
```


Code > Matrix Multiplication



The screenshot displays the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor window shows a script with a custom function named `matrix_multiplication`. The function takes two matrices, `mat1` and `mat2`, and checks if the number of columns in `mat1` is equal to the number of rows in `mat2`. If not, it stops with an error message. If they are compatible, it calculates the matrix product using `%*%` and returns the result.

```
1 matrix_multiplication <- function(mat1, mat2) {  
2   if (ncol(mat1) != nrow(mat2)) {  
3     stop("Number of columns in the first matrix must be equal to the number of rows in the second matrix")  
4   }  
5  
6   result <- mat1 %*% mat2  
7   return(result)  
8 }  
9
```

The console window at the bottom shows the execution of the function with two example matrices:

```
>  
> m1 <- matrix(c(1,2,3,4),nrow = 2, ncol = 2)  
> m2 <- matrix(c(2,3,4,5),nrow = 2,ncol = 2)  
> matrix_multiplication(m1, m2)  
      [,1] [,2]  
[1,]   11  19  
[2,]   16  28  
>
```