ASSIGNMENT - 1

Name :- Mohit Balachander
Regd No:- 22MIC7042

Date:- 16-2-24

Slot - L17-L18

---

1. Sum of all digits of a given number.

 # Function to calculate the sum of digits

```
n <- 96543

sum <- 0

while (n > 0) {

   temp <- n %% 10

   sum <- sum + temp

   n <- floor(n / 10)

}

sum
```

OUTPUT:-

```
R Console
> # Function to calculate the sum of digits
>
> n <- 35766786
> sum <- 0
> while (n > 0) {
+     temp <- n %% 10
+     sum <- sum + temp
+     n <- floor(n / 10)
+ }
> sum
[1] 48
>
```

---

2. To check whether the given number is Prime Nmber or not.

 # To check whether the given number is Prime Nmber or not

```
Prime_No <- function(n1) {
if (n1 == 2) {
        return(TRUE)
}
if (n1 <= 1) {
        return(FALSE)
}
for (i in 2:(n1-1)) {
        if (n1 %% i == 0) {
        return(FALSE)
        }
}
return(TRUE)
```

```
}

num <- 13

if (Prime_No(num)) {

print(paste(num, "is a prime number"))

} else {

print("It is not a prime number")

}
```

OUTPUT:-

```
R Console

> Prime_No <- function(n1) {
+ if (n1 == 2) {
+ return(TRUE)
+ }
+ if (n1 <= 1) {
+ return(FALSE)
+ }
+ for (i in 2:(n1-1)) {
+ if (n1 %% i == 0) {
+ return(FALSE)
+ }
+ }
+ return(TRUE)
+ }
> num <- 13
> if (Prime_No(num)) {
+ print(paste(num, "is a prime number"))
+ } else {
+ print("It is not a prime number")
+ }
[1] "13 is a prime number"
>
```

---

3. Write a program in R to find prime number within a range.

 # Function to check if a number is prime

```
prime <- function(n) {
```

```
  if (n <= 1) return(FALSE)
  for (i in 2:sqrt(n)) if (n %% i == 0) return(FALSE)
  TRUE
}
primes_in_range <- function(start, end) {
  primes <- c()
  for (num in start:end) {
    if (prime(num)) primes <- c(primes, num)
  }
  primes
}
start <- 1
end <- 50
prime_num <- find_primes_in_range(start, end)
cat("Prime numbers within the range", start, "to", end, "are:", prime_num, "\n")
```

OUTPUT:-

```
R Console
> # Function to check if a number is prime
> prime <- function(n) {
+   if (n <= 1) return(FALSE)
+   for (i in 2:sqrt(n)) if (n %% i == 0) return(FALSE)
+   TRUE
+ }
> primes_in_range <- function(start, end) {
+   primes <- c()
+   for (num in start:end) {
+     if (prime(num)) primes <- c(primes, num)
+   }
+   primes
+ }
>
> start <- 1
> end <- 50
> prime_num <- find_primes_in_range(start, end)
>
> cat("Prime numbers within the range", start, "to", end, "are:", prime_num, "\n")
Prime numbers within the range 1 to 50 are: 3 5 7 11 13 17 19 23 29 31 37 41 43 47
>
```

---

4. Write a program in R to find the prefect number.

#to find the prefect number from 1:100

```
for (k in 1:100) {
   n = k
   i = 1
   s = 0

   while (i < n) {
     if (n %% i == 0) {
       s = s + i
     }
     i = i + 1
   }
   if (s == n) {
     print(paste(n,"is a perfect number"))
   } else{
     print(paste(n,"is not a perfect number"))
   }
k=k+1
}
```

OUTPUT:-

R Console

```
[1] "1 is not a perfect number"
[1] "2 is not a perfect number"
[1] "3 is not a perfect number"
[1] "4 is not a perfect number"
[1] "5 is not a perfect number"
[1] "6 is a perfect number"
[1] "7 is not a perfect number"
[1] "8 is not a perfect number"
[1] "9 is not a perfect number"
[1] "10 is not a perfect number"
[1] "11 is not a perfect number"
[1] "12 is not a perfect number"
[1] "13 is not a perfect number"
[1] "14 is not a perfect number"
[1] "15 is not a perfect number"
[1] "16 is not a perfect number"
[1] "17 is not a perfect number"
[1] "18 is not a perfect number"
[1] "19 is not a perfect number"
[1] "20 is not a perfect number"
[1] "21 is not a perfect number"
[1] "22 is not a perfect number"
[1] "23 is not a perfect number"
[1] "24 is not a perfect number"
[1] "25 is not a perfect number"
[1] "26 is not a perfect number"
[1] "27 is not a perfect number"
[1] "28 is a perfect number"
[1] "29 is not a perfect number"
[1] "30 is not a perfect number"
[1] "31 is not a perfect number"
[1] "32 is not a perfect number"
[1] "33 is not a perfect number"
[1] "34 is not a perfect number"
[1] "35 is not a perfect number"
```

R Console

```
[1] "35 is not a perfect number"
[1] "36 is not a perfect number"
[1] "37 is not a perfect number"
[1] "38 is not a perfect number"
[1] "39 is not a perfect number"
[1] "40 is not a perfect number"
[1] "41 is not a perfect number"
[1] "42 is not a perfect number"
[1] "43 is not a perfect number"
[1] "44 is not a perfect number"
[1] "45 is not a perfect number"
[1] "46 is not a perfect number"
[1] "47 is not a perfect number"
[1] "48 is not a perfect number"
[1] "49 is not a perfect number"
[1] "50 is not a perfect number"
[1] "51 is not a perfect number"
[1] "52 is not a perfect number"
[1] "53 is not a perfect number"
[1] "54 is not a perfect number"
[1] "55 is not a perfect number"
[1] "56 is not a perfect number"
[1] "57 is not a perfect number"
[1] "58 is not a perfect number"
[1] "59 is not a perfect number"
[1] "60 is not a perfect number"
[1] "61 is not a perfect number"
[1] "62 is not a perfect number"
[1] "63 is not a perfect number"
[1] "64 is not a perfect number"
[1] "65 is not a perfect number"
[1] "66 is not a perfect number"
[1] "67 is not a perfect number"
[1] "68 is not a perfect number"
[1] "69 is not a perfect number"
```

```
R Console
[1] "67 is not a perfect number"
[1] "68 is not a perfect number"
[1] "69 is not a perfect number"
[1] "70 is not a perfect number"
[1] "71 is not a perfect number"
[1] "72 is not a perfect number"
[1] "73 is not a perfect number"
[1] "74 is not a perfect number"
[1] "75 is not a perfect number"
[1] "76 is not a perfect number"
[1] "77 is not a perfect number"
[1] "78 is not a perfect number"
[1] "79 is not a perfect number"
[1] "80 is not a perfect number"
[1] "81 is not a perfect number"
[1] "82 is not a perfect number"
[1] "83 is not a perfect number"
[1] "84 is not a perfect number"
[1] "85 is not a perfect number"
[1] "86 is not a perfect number"
[1] "87 is not a perfect number"
[1] "88 is not a perfect number"
[1] "89 is not a perfect number"
[1] "90 is not a perfect number"
[1] "91 is not a perfect number"
[1] "92 is not a perfect number"
[1] "93 is not a perfect number"
[1] "94 is not a perfect number"
[1] "95 is not a perfect number"
[1] "96 is not a perfect number"
[1] "97 is not a perfect number"
[1] "98 is not a perfect number"
[1] "99 is not a perfect number"
[1] "100 is not a perfect number"
>
```

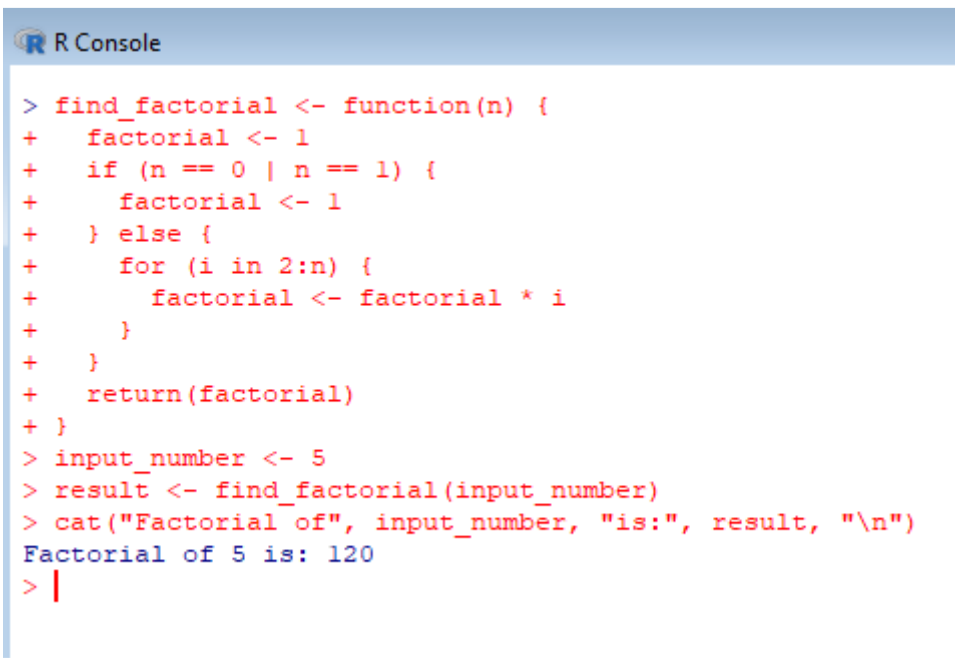
5.Write a program in R to find the factorial of a number.

 # to find factorial of a number

```
find_factorial <- function(n) {
  factorial <- 1
  if (n == 0 | n == 1) {
    factorial <- 1
  } else {
```

```
  for (i in 2:n) {
    factorial <- factorial * i
  }
 }
 return(factorial)
}
input_number <- 5
result <- find_factorial(input_number)
cat("Factorial of", input_number, "is:", result, "\n")
```

OUTPUT:-



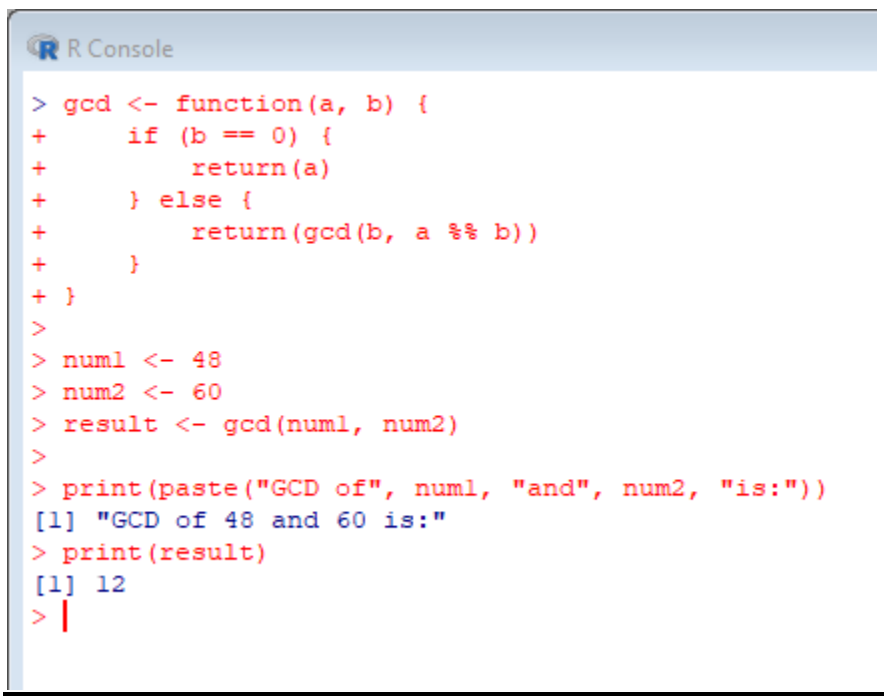
---

6. To find GCD of given two number.

#to find GCD of given two number

```
gcd <- function(a, b) {
  if (b == 0) {
    return(a)
  } else {
    return(gcd(b, a %% b))
  }
}
num1 <- 48
num2 <- 60
result <- gcd(num1, num2)
print(paste("GCD of", num1, "and", num2, "is:"))
print(result)
```

OUTPUT:-

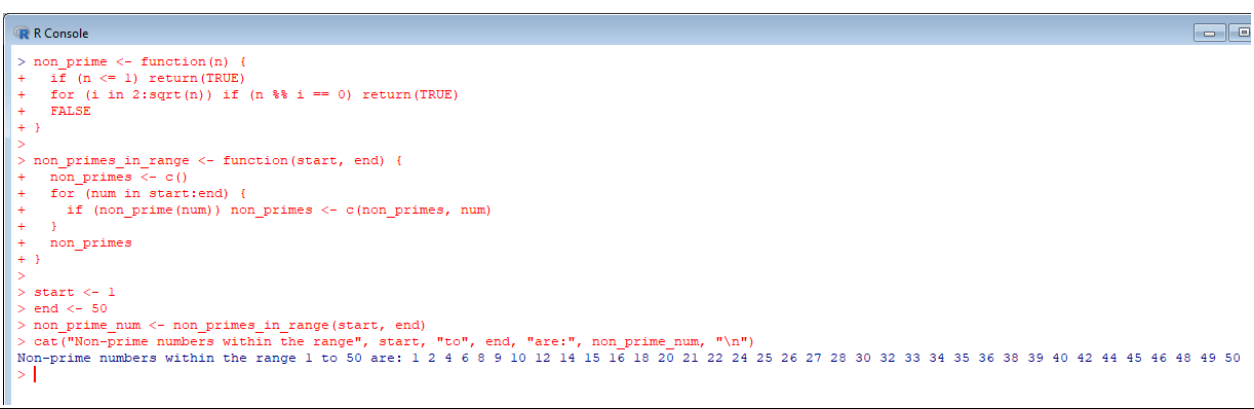


7. To print all non prime numbers from 1-N

#to print all non-prime numbers from 1-N

```
non_prime <- function(n) {
  if (n <= 1) return(TRUE)
  for (i in 2:sqrt(n)) if (n %% i == 0) return(TRUE)
  FALSE
}
non_primes_in_range <- function(start, end) {
  non_primes <- c()
  for (num in start:end) {
    if (non_prime(num)) non_primes <- c(non_primes, num)
  }
  non_primes
}
start <- 1
end <- 50
non_prime_num <- non_primes_in_range(start, end)
cat("Non-prime numbers within the range", start, "to", end, "are:", non_prime_num, "\n")
```

OUTPUT:-



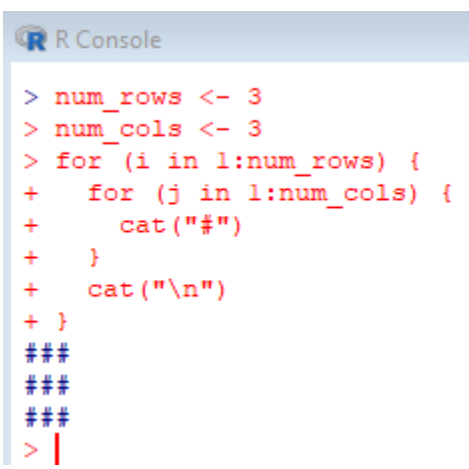
8. Write a program in R to print below pattern

```
# # #
# # #
# # #
```

#print the above pattern

```
num_rows <- 3
num_cols <- 3
for (i in 1:num_rows) {
  for (j in 1:num_cols) {
    cat("#")
  }
  cat("\n")
}
```

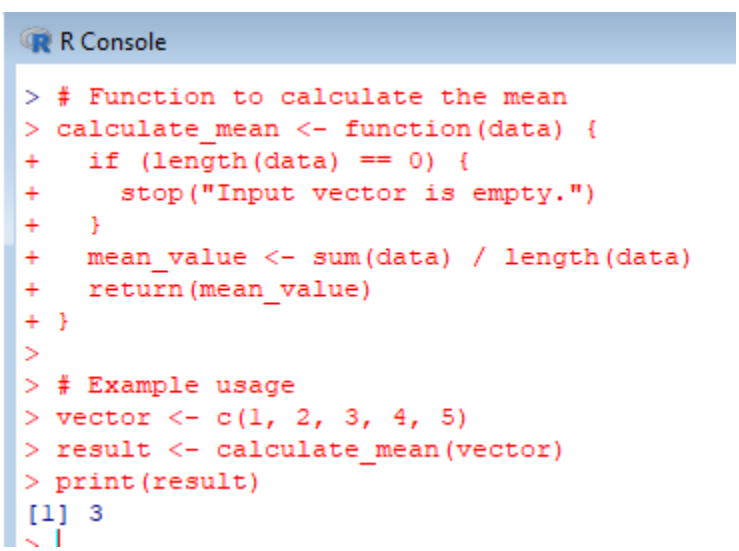
OUTPUT:-



9. Write a R program to find the mean of the number.

```
#program to find mean of the number

calculate_mean <- function(data) {

  if (length(data) == 0) {

    stop("Input vector is empty.")

  }

  mean_value <- sum(data) / length(data)

  return(mean_value)

}

vector <- c(1, 2, 3, 4, 5)

result <- calculate_mean(vector)

print(result)
```

OUTPUT:



---

10. Write a R program to find the median of the number.

```
#program to find the median of the number
# Function to calculate the median
calculate_median <- function(data) {
  if (length(data) == 0) {
    stop("Input vector is empty.")
  }
  sorted_data <- sort(data)
  n <- length(sorted_data)
  if (n %% 2 == 0) {
    median_value <- (sorted_data[n/2] + sorted_data[(n/2) + 1]) / 2
  } else {
    median_value <- sorted_data[(n + 1) / 2]
  }
  return(median_value)
}
vector <- c(1, 3, 2, 5, 4)
result <- calculate_median(vector)
print(result)
```

OUTPUT:

```
R Console
> # Function to calculate the median
> calculate_median <- function(data) {
+   if (length(data) == 0) {
+     stop("Input vector is empty.")
+   }
+   sorted_data <- sort(data)
+   n <- length(sorted_data)
+   if (n %% 2 == 0) {
+     median_value <- (sorted_data[n/2] + sorted_data[(n/2) + 1]) / 2
+   } else {
+     median_value <- sorted_data[(n + 1) / 2]
+   }
+   return(median_value)
+ }
>
> vector <- c(1, 3, 2, 5, 4)
> result <- calculate_median(vector)
> print(result)
[1] 3
>
```

---

11. Write a R program to find mode of the number.

```
mode <- function(x) {
  uniqx <- unique(x)
  freq <- tabulate(match(x, uniqx))
  modes <- uniqx[freq == max(freq)]
  if (length(modes) == length(uniqx)) {
    cat("No mode found\n")
  } else {
    cat("Mode(s):", modes, "\n")
  }
}
my_array <- c(1, 2, 3, 2, 4, 5, 3, 3)
mode(my_array)
```

OUTPUT:

```
R Console
> mode <- function(x) {
+   uniqx <- unique(x)
+   freq <- tabulate(match(x, uniqx))
+   modes <- uniqx[freq == max(freq)]
+
+   if (length(modes) == length(uniqx)) {
+     cat("No mode found\n")
+   } else {
+     cat("Mode(s):", modes, "\n")
+   }
+ }
> my_array <- c(1, 2, 3, 2, 4, 5, 3, 3)
> mode(my_array)
Mode(s): 3
>
```

---

12. Write a R program to find the standard deviation of the number.

```
sd <- function(data) {
  if (length(data) == 0) {
    stop("Input vector is empty.")
  }
  mean_value <- mean(data)
  variance <- sum((data - mean_value)^2) / length(data)
  standard_deviation <- sqrt(variance)
  return(standard_deviation)
}
vector <- c(1, 3, 2, 5, 4)
result <- sd(vector)
print(result)
```

OUTPUT:

```
R Console
> sd <- function(data) {
+   if (length(data) == 0) {
+     stop("Input vector is empty.")
+   }
+   mean_value <- mean(data)
+   variance <- sum((data - mean_value)^2) / length(data)
+   standard_deviation <- sqrt(variance)
+   return(standard_deviation)
+ }
> vector <- c(1, 3, 2, 5, 4)
> result <- sd(vector)
> print(result)
[1] 1.414214
```

---

13. Write a R program to find all the natural number from 1 to 10 except 5.

```
#program to find all the natural numbers from 1 to 10 except 5
numbers <- vector()
for (i in 1:10) {
  # check if the current number is 5
  if (i == 5) {
    next
  }
  numbers <- c(numbers, i)
}
print(numbers)
```

OUTPUT:

R Console

```
> numbers <- vector()
> for (i in 1:10) {
+   # check if the current number is 5
+   if (i == 5) {
+     next
+   }
+   numbers <- c(numbers, i)
+ }
> print(numbers)
[1]  1  2  3  4  6  7  8  9 10
>
```

ASSIGNMENT-2

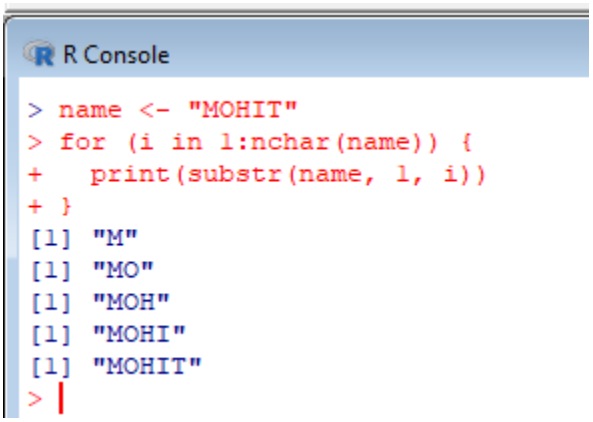
Name:- Mohit Balachander
Regd No:- 22MIC7042
Slot:- L17+L18

1.Write a R program to print the name MOHIT

M
MO
MOH
MOHI
MOHIT

```
name <- "MOHIT"
for (i in 1:nchar(name)) {
  print(substr(name, 1, i))
}
```

OUTPUT:-



2.Write a R program to find the maximum and the minimum value of a given vector

```
find_max <- function(a) {
  b <- length(a)
  max_val <- a[[1]]
  for (i in 1:b) {
    if (a[[i]] > max_val) {
      max_val <- a[[i]]
    }
  }
  return(max_val)
```

```
}
find_min <- function(a) {
  b <- length(a)
  min_val <- a[[1]]
  for (i in 1:b) {
    if (a[[i]] < min_val) {
      min_val <- a[[i]]
    }
  }
  return(min_val)
}
ab <- list(1, 23, 32, 44, 69, 96)
max_value <- find_max(ab)
min_value <- find_min(ab)
print(paste("Maximum value:", max_value))
print(paste("Minimum value:", min_value))
```

OUTPUT:

```
R Console
> find_max <- function(a) {
+   b <- length(a)
+   max_val <- a[[1]]
+   for (i in 1:b) {
+     if (a[[i]] > max_val) {
+       max_val <- a[[i]]
+     }
+   }
+   return(max_val)
+ }
>
> find_min <- function(a) {
+   b <- length(a)
+   min_val <- a[[1]]
+   for (i in 1:b) {
+     if (a[[i]] < min_val) {
+       min_val <- a[[i]]
+     }
+   }
+   return(min_val)
+ }
>
> ab <- list(1, 23, 32, 44, 69, 96)
> max_value <- find_max(ab)
> min_value <- find_min(ab)
> print(paste("Maximum value:", max_value))
[1] "Maximum value: 96"
> print(paste("Minimum value:", min_value))
[1] "Minimum value: 1"
>
```

3. Write a program in R to display the cube of the number up to given an integer.

```
cubes <- function(n) {
  if (n <= 0) {
    stop("Input must be a positive integer.")
  }
  for (i in 1:n) {
    cube <- i^3
    cat("Cube of", i, "is", cube, "\n")
  }
}
cubes(5)
```

OUTPUT:

```
R Console

> cubes <- function(n) {
+   if (n <= 0) {
+     stop("Input must be a positive integer.")
+   }
+   for (i in 1:n) {
+     cube <- i^3
+     cat("Cube of", i, "is", cube, "\n")
+   }
+ }
> cubes(5)
Cube of 1 is 1
Cube of 2 is 8
Cube of 3 is 27
Cube of 4 is 64
Cube of 5 is 125
>
```

---

4. Write a program in R to display the first n terms of Fibonacci series.

Sample Output:

Input number of terms to display: 10

Here is the Fibonacci series upto to 10 terms:

0 1 1 2 3 5 8 13 21 34

```
# Function to print the Fibonacci sequence using a loop
print_fibonacci <- function(n) {
a <- 0
b <- 1

cat("Fibonacci Sequence:")
```

```
for (i in 1:n) {
        cat(a, " ")
        next_num <- a + b
        a <- b
        b <- next_num
}
}
number_of_terms <- 10
print_fibonacci(number_of_terms)
```

OUTPUT:

```
> print_fibonacci <- function(n) {
+ a <- 0
+ b <- 1
+
+ cat("Fibonacci Sequence:")
+ for (i in 1:n) {
+ cat(a, " ")
+ next_num <- a + b
+ a <- b
+ b <- next_num
+ }
+ }
> number_of_terms <- 10
> print_fibonacci(number_of_terms)
Fibonacci Sequence:0  1  1  2  3  5  8  13  21  34  >
```

---

5. Write a program in R to display the number in reverse order.

Sample Output:

Input a number: 12345

The number in reverse order is : 54321

```
{
  n = as.integer(readline(prompt = "Enter a number :"))
  rev = 0
  while (n > 0) {
    r = n %% 10
    rev = rev * 10 + r
    n = n %/% 10
  }
  print(paste("Reverse number is :", rev))
}
```

OUTPUT:

```
R Console
> {
+     n = as.integer(readline(prompt = "Enter a number :"))
+     rev = 0
+
+     while (n > 0) {
+       r = n %% 10
+       rev = rev * 10 + r
+       n = n %/% 10
+     }
+
+     print(paste("Reverse number is :", rev))
+ }
Enter a number :12345
[1] "Reverse number is : 54321"
>
```

---

6. Write a program in R to find the length of a string without using the library function.

```
input_string <- readline(prompt="Input a string: ")
string_length <- function(input_string) {
 count <- 0
 for (char in strsplit(input_string, NULL)[[1]]) {
 count <- count + 1
 }
 return(count)
}
length_result <- string_length(input_string)
cat("The string contains", length_result, "number of characters.\n")
cat("So, the length of the string", input_string, "is:", length_result, "\n")
```

OUTPUT:

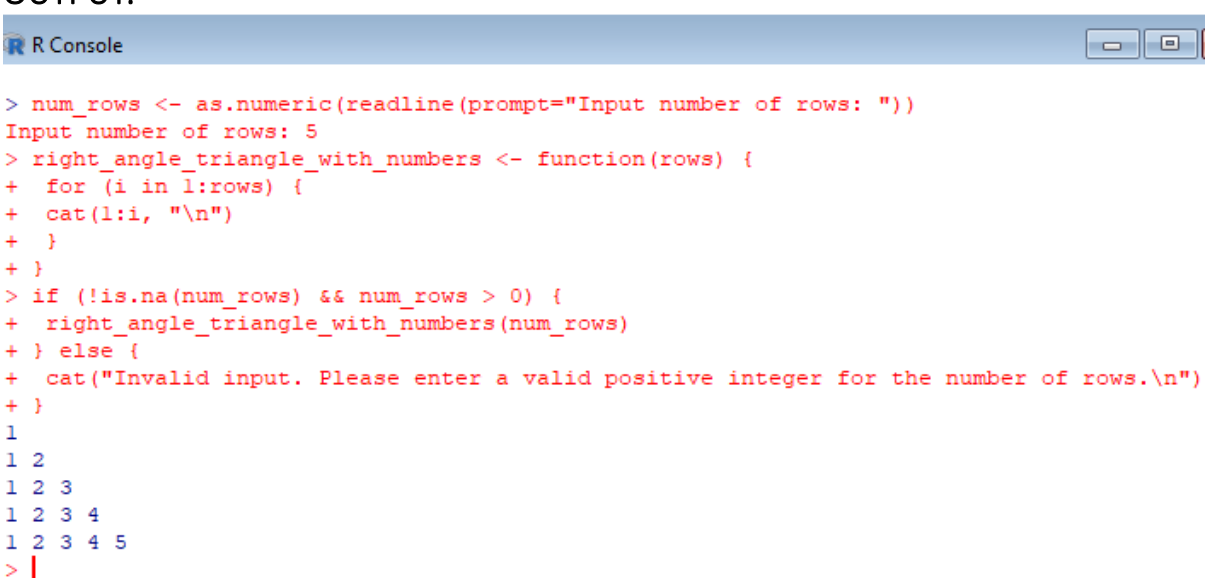
```
R Console
> input_string <- readline(prompt="Input a string: ")
Input a string: MOHIT
> string_length <- function(input_string) {
+  count <- 0
+  for (char in strsplit(input_string, NULL)[[1]]) {
+  count <- count + 1
+  }
+  return(count)
+ }
> length_result <- string_length(input_string)
> cat("The string contains", length_result, "number of characters.\n")
The string contains 5 number of characters.
> cat("So, the length of the string", input_string, "is:", length_result, "\n")
So, the length of the string MOHIT is: 5
```

---

7. Write a program in R to display the pattern like right angle triangle with number.
Sample Output: Input number of rows: 5
1
12
123
1234
12345

```
num_rows <- as.numeric(readline(prompt="Input number of rows: "))
right_angle_triangle_with_numbers <- function(rows) {
 for (i in 1:rows) {
 cat(1:i, "\n")
 }
}
if (!is.na(num_rows) && num_rows > 0) {
 right_angle_triangle_with_numbers(num_rows)
} else {
 cat("Invalid input. Please enter a valid positive integer for the number of rows.\n")
}
```

OUTPUT:



---

8. Write a program in R to make such a pattern like right angle triangle with number increased by 1.

Sample Output:

Input number of rows: 4

1

2 3

4 5 6

7 8 9 10

```
num_rows <- as.numeric(readline(prompt="Input number of rows: "))
triangle <- function(rows) {
 count <- 1
 for (i in 1:rows) {
 cat(seq(count, count + i- 1), "\n")
 count <- count + i
 }
}
if (!is.na(num_rows) && num_rows > 0) {
 triangle(num_rows)
} else {
 cat("Invalid input. Please enter a valid positive integer for the number of rows.\n")
}
```

OUTPUT:-

```
> num_rows <- as.numeric(readline(prompt="Input number of rows: "))
Input number of rows: 4
> triangle <- function(rows) {
+  count <- 1
+  for (i in 1:rows) {
+  cat(seq(count, count + i - 1), "\n")
+  count <- count + i
+  }
+ }
> if (!is.na(num_rows) && num_rows > 0) {
+  triangle(num_rows)
+ } else {
+  cat("Invalid input. Please enter a valid positive integer for the number of rows.\n")
+ }
1
2 3
4 5 6
7 8 9 10
>
```