Introduction to R Programming Lab

(BTCCSPCP501)

An Assignment Work Submitted for the 5th Semester
of Bachelor of Technology

in

Computer Science and Engineering (Data Science)

by

Suman Mondal

Registration Number: 100227240046

DEPARTMENT OF COMPUTER SCIENCE
KAZI NAZRUL UNIVERSITY
ASANSOL - 713340, WEST BENGAL

Contents

Chapter 1

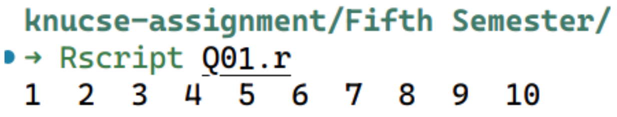
Assignment 3

1.1 Write a R program to print all natural numbers from 1 to n

Source Code :

```r
# Q01. Write a R program to print all natural numbers from 1 to n

print_natural_numbers <- function(n) {
  if (n <= 0) {
    cat("Please provide a positive integer value for n.\n")
    return(NULL)
  }

  for (i in 1:n) {
    cat(i, " ")
  }
  cat("\n")
}

print_natural_numbers(10)
```

Program Output :



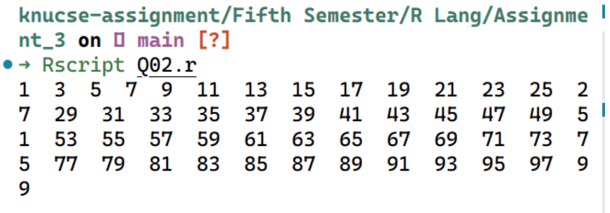
```
knucse-assignment/Fifth Semester/
→ Rscript Q01.r
1  2  3  4  5  6  7  8  9  10
```

1.2 Write a R program to print all odd number between 1 to 100

Source Code :

```
# Q02. Write a R program to print all odd number between 1 to
↪  100

print_odd_numbers <- function() {
  for (i in 1:100) {
    if (i %% 2 != 0) {
      cat(i, " ")
    }
  }
  cat("\n")
}

print_odd_numbers()
```

Program Output :



```
knucse-assignment/Fifth Semester/R Lang/Assignme
nt_3 on  main [?]
● → Rscript Q02.r
1  3  5  7  9  11  13  15  17  19  21  23  25  2
7  29  31  33  35  37  39  41  43  45  47  49  5
1  53  55  57  59  61  63  65  67  69  71  73  7
5  77  79  81  83  85  87  89  91  93  95  97  9
9
```

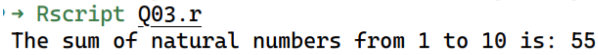
1.3 Write a R program to find sum of all natural numbers between 1 to n

Source Code :

```
# Q03. Write a R program to find sum of all natural numbers
↪  between 1 to n

sum_of_natural_numbers <- function(n) {
  if (n <= 0) {
    cat("Please provide a positive integer value for n.\n")
    return(NULL)
  }

  sum <- 0
  for (i in 1:n) {
    sum <- sum + i
  }

  return(sum)
```

```r
}

n <- 10
result <- sum_of_natural_numbers(n)
cat("The sum of natural numbers from 1 to", n, "is:", result, "\n")
```

Program Output :



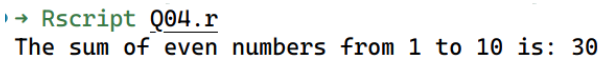
```
→ Rscript Q03.r
 The sum of natural numbers from 1 to 10 is: 55
```

1.4 Write a R program to find sum of all even numbers between 1 to n

Source Code :

```r
# Q04. Write a R program to find sum of all even numbers between 1 to n

sum_of_even_numbers <- function(n) {
  if (n <= 0) {
    cat("Please provide a positive integer value for n.\n")
    return(NULL)
  }

  sum <- 0
  for (i in 2:n) {
    if (i %% 2 == 0) {
      sum <- sum + i
    }
  }

  return(sum)
}

n <- 10
result <- sum_of_even_numbers(n)
cat("The sum of even numbers from 1 to", n, "is:", result, "\n")
```

Program Output :



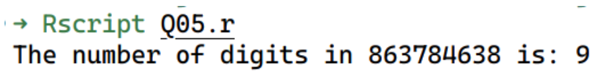
```
→ Rscript Q04.r
The sum of even numbers from 1 to 10 is: 30
```

1.5 Write a R program to count number of digits in a number

Source Code :

```r
# Q05. Write a R program to count number of digits in a number

count_digits <- function(number) {
  if (number < 0) {
    number <- -number
  }

  num_digits <- 0
  while (number > 0) {
    number <- number %/% 10
    num_digits <- num_digits + 1
  }

  return(num_digits)
}

number <- 863784638
num_digits <- count_digits(number)
cat("The number of digits in", number, "is:", num_digits, "\n")
```

Program Output :



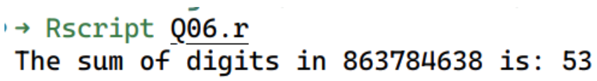
```
→ Rscript Q05.r
The number of digits in 863784638 is: 9
```

1.6 Write a R program to calculate sum of digits of a number

Source Code :

```r
# Q06. Write a R program to calculate sum of digits of a number

sum_of_digits <- function(number) {
  if (number < 0) {
    number <- -number
  }
  sum <- 0
  while (number > 0) {
    digit <- number %% 10
    sum <- sum + digit
    number <- number %/% 10
  }

  return(sum)
}

number <- 863784638
digit_sum <- sum_of_digits(number)
cat("The sum of digits in", number, "is:", digit_sum, "\n")
```

Program Output :



```
→ Rscript Q06.r
The sum of digits in 863784638 is: 53
```

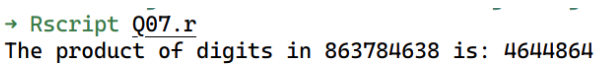
1.7 Write a R program to calculate product of digits of a number

Source Code :

```
# Q07. Write a R program to calculate product of digits of a
↪   number

product_of_digits <- function(number) {
  if (number < 0) {
    number <- -number
  }

  product <- 1
  while (number > 0) {
    digit <- number %% 10
    product <- product * digit
    number <- number %/% 10
  }

  return(product)
}
```

```r
number <- 863784638
digit_product <- product_of_digits(number)
cat("The product of digits in", number, "is:", digit_product, "\n")
```

Program Output :



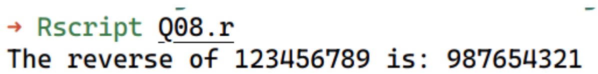
```
→ Rscript Q07.r
The product of digits in 863784638 is: 4644864
```

1.8 Write a R program to enter a number and print its reverse

Source Code :

```r
# Q08. Write a R program to enter a number and print its reverse

reverse_number <- function(number) {
  reversed <- 0
  while (number > 0) {
    digit <- number %% 10
    reversed <- reversed * 10 + digit
    number <- number %/% 10
  }

  return(reversed)
}

# Take user input for the number
number <- 123456789
reversed_number <- reverse_number(number)
cat("The reverse of", number, "is:", reversed_number, "\n")
```

Program Output :



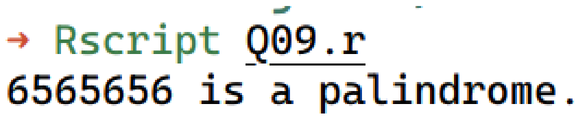
```
→ Rscript Q08.r
The reverse of 123456789 is: 987654321
```

1.9 Write a R program to check whether a number is palindrome or not

Source Code :

```r
# Q09. Write a R program to check whether a number is
↪  palindrome or not

is_palindrome <- function(number) {
  original <- number
  reversed <- 0

  while (number > 0) {
    digit <- number %% 10
    reversed <- reversed * 10 + digit
    number <- number %/% 10
  }

  return(original == reversed)
}

# Take user input for the number
number <- 6565656
if (is_palindrome(number)) {
  cat(number, "is a palindrome.\n")
} else {
  cat(number, "is not a palindrome.\n")
}
```

Program Output :



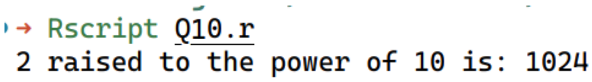
```
→ Rscript Q09.r
6565656 is a palindrome.
```

1.10 Write a R program to find power of a number using for loop

Source Code :

```r
# Q10. Write a R program to find power of a number using for
↪  loop

calculate_power <- function(base, exponent) {
  result <- 1

  for (i in 1:exponent) {
    result <- result * base
  }

  return(result)
}

base <- 2
exponent <- 10
power_result <- calculate_power(base, exponent)
cat(base, "raised to the power of", exponent, "is:", power_result, "\n")
```

Program Output :



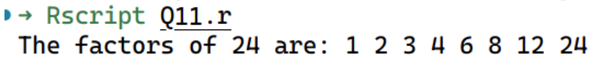
```
→ Rscript Q10.r
2 raised to the power of 10 is: 1024
```

1.11 Write a R program to find all factors of a number

Source Code :

```r
# Q11. Write a R program to find all factors of a number

find_factors <- function(number) {
  factors <- c()

  for (i in 1:number) {
    if (number %% i == 0) {
      factors <- c(factors, i)
    }
  }

  return(factors)
}

number <- 24
factor_list <- find_factors(number)
cat("The factors of", number, "are:", factor_list, "\n")
```

Program Output :



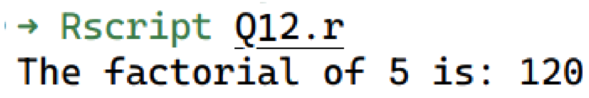
```
→ Rscript Q11.r
The factors of 24 are: 1 2 3 4 6 8 12 24
```

1.12 Write a R program to calculate factorial of a number

Source Code :

```
# Q12. Write a R program to calculate factorial of a number

calculate_factorial <- function(number) {
  if (number < 0) {
    cat("Factorial is not defined for negative numbers.\n")
    return(NULL)
  }

  factorial <- 1
  for (i in 1:number) {
    factorial <- factorial * i
  }

  return(factorial)
}

number <- 5
factorial_result <- calculate_factorial(number)

if (!is.null(factorial_result)) {
  cat("The factorial of", number, "is:", factorial_result, "\n")
}
```

Program Output :



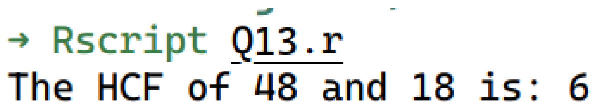
```
→ Rscript Q12.r
The factorial of 5 is: 120
```

1.13 Write a R program to find HCF (GCD) of two numbers

Source Code :

```
# Q13. Write a R program to find HRF (GRD) of two numbers

calculate_hcf <- function(a, b) {
  while (b != 0) {
    temp <- b
    b <- a %% b
    a <- temp
  }
  return(a)
}

number1 <- 48
number2 <- 18

hcf_result <- calculate_hcf(number1, number2)
cat("The HCF of", number1, "and", number2, "is:", hcf_result,
 ↪  "\n")
```

Program Output :



```
→ Rscript Q13.r
The HCF of 48 and 18 is: 6
```

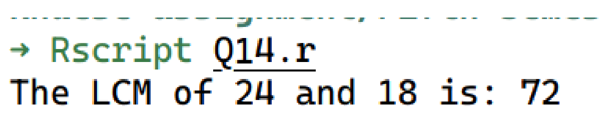
1.14 Write a R program to find LCM of two numbers

Source Code :

```
# Q14. Write a R program to find LRM of two numbers

calculate_gcd <- function(a, b) {
  while (b != 0) {
    temp <- b
    b <- a %% b
    a <- temp
  }
  return(a)
}

calculate_lcm <- function(a, b) {
  gcd <- calculate_gcd(a, b)
  lcm <- (a * b) / gcd
  return(lcm)
}

number1 <- 24
number2 <- 18
```

```
lcm_result <- calculate_lcm(number1, number2)
cat("The LCM of", number1, "and", number2, "is:", lcm_result,
 ↪  "\n")
```

Program Output :



```
➜ Rscript Q14.r
The LCM of 24 and 18 is: 72
```

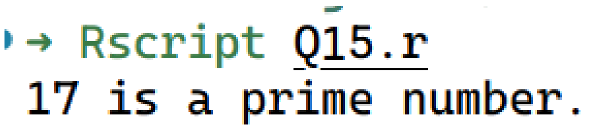
1.15 Write a R program to check whether a number is Prime number or not

Source Code :

```
# Q15. Write a R program to check whether a number is Prime
 ↪  number or not

is_prime <- function(number) {
  if (number <= 1) {
    return(FALSE)
  }

  if (number <= 3) {
    return(TRUE)
  }

  if (number %% 2 == 0 || number %% 3 == 0) {
    return(FALSE)
  }

  i <- 5
  while (i * i <= number) {
    if (number %% i == 0 || number %% (i + 2) == 0) {
      return(FALSE)
    }
    i <- i + 6
  }

  return(TRUE)
}
```

```r
number <- 17

if (is_prime(number)) {
  cat(number, "is a prime number.\n")
} else {
  cat(number, "is not a prime number.\n")
}
```

Program Output :



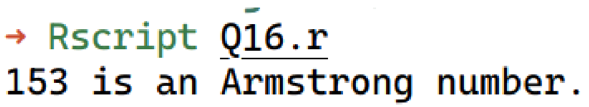
```
→ Rscript Q15.r
 17 is a prime number.
```

1.16 Write a R program to check whether a number is Armstrong number or not

Source Code :

```r
# Q16. Write a R program to check whether a number is Armstrong
↪   number or not

is_armstrong <- function(number) {
  num_copy <- number
  num_digits <- nchar(number)
  armstrong_sum <- 0

  while (num_copy > 0) {
    digit <- num_copy %% 10
    armstrong_sum <- armstrong_sum + digit ^ num_digits
    num_copy <- num_copy %/% 10
  }

  return(armstrong_sum == number)
}

number <- 153

if (is_armstrong(number)) {
  cat(number, "is an Armstrong number.\n")
} else {
  cat(number, "is not an Armstrong number.\n")
}
```

Program Output :



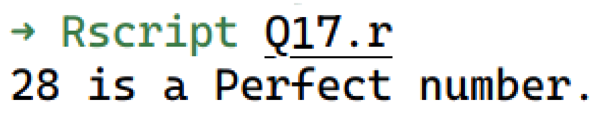
```
→ Rscript Q16.r
153 is an Armstrong number.
```

1.17 Write a R program to check whether a number is Perfect number or not

Source Code :

```
# Q17. Write a R program to check whether a number is Perfect number or not

is_perfect <- function(number) {
  if (number <= 0) {
    return(FALSE)
  }

  divisors_sum <- 0
  for (i in 1:(number/2)) {
    if (number %% i == 0) {
      divisors_sum <- divisors_sum + i
    }
  }

  return(divisors_sum == number)
}

number <- 28

if (is_perfect(number)) {
  cat(number, "is a Perfect number.\n")
} else {
  cat(number, "is not a Perfect number.\n")
}
```

Program Output :



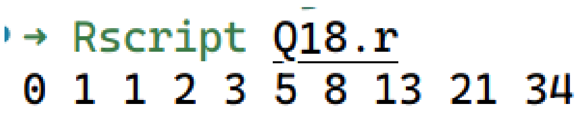
```
→ Rscript Q17.r
28 is a Perfect number.
```

1.18 Write a R program to print Fibonacci series up to n terms

Source Code :

```r
# Q18. Write a R program to print Fibonacci series up to n terms

print_fibonacci <- function(n) {
  if (n <= 0) {
    cat("Please provide a positive integer value for n.\n")
    return(NULL)
  }

  fib_series <- c(0, 1)

  if (n == 1) {
    cat(fib_series[1], "\n")
  } else if (n == 2) {
    cat(fib_series, "\n")
  } else {
    for (i in 3:n) {
      next_term <- fib_series[i - 1] + fib_series[i - 2]
      fib_series <- c(fib_series, next_term)
    }
    cat(fib_series, "\n")
  }
}

n <- 10
print_fibonacci(n)
```

Program Output :



```
→ Rscript Q18.r
0 1 1 2 3 5 8 13 21 34
```