**R PROGRAMS:**

**Q1. The prime factors of 13195 are 5, 7, 13 and 29. What is the largest prime factor of the number?**

prime.factors <- function (n) {

factors <- c() # Define list of factors

primes <- esieve(floor(sqrt(n))) # Define primes to be tested

d <- which(n%%primes == 0) # Idenitfy prime divisors

if (length(d) == 0) # No prime divisors

return(n)

for (q in primes[d]) { # Test candidate primes

while (n%%q == 0) { # Generate list of factors

factors <- c(factors, q)

n <- n/q } } if (n > 1) factors <- c(factors, n)

return(factors)

}

max(prime.factors(600851475143))

**Q2. The sum of the squares of the first ten natural numbers is,**

**12+22+...+102=385**

**The square of the sum of the first ten natural numbers is,**

**(1+2+...+10)2=552=3025**

**Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is**

**3025−385=2640**

**.Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.**

new.function <- function(n) {

b <- (n\*(n+1)(2\*n+1)/6 - (n(n+1)/2)\*\*2)

print(b)

}

new.function(100)

**Q3. A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009 = 91 × 99.**

**Find the largest palindrome made from the product of two 3-digit numbers.**

function is\_palindrome(initial\_number) {

console.log('initial\_number:',initial\_number)

var reversed = 0;

var temp = initial\_number;

while (temp > 0) {

var last\_digit = temp % 10; // extract the last digit

reversed = reversed \* 10 + last\_digit; // add last digit

temp = parseInt(temp / 10); // remove last digit

console.log(temp, reversed)

}

console.log('initial\_number === reversed', initial\_number, reversed, initial\_number === reversed)

console.log()

return initial\_number === reversed;

}

is\_palindrome(9009)

is\_palindrome(123321)

is\_palindrome(123456)

**Q4. The following iterative sequence is defined for the set of positive integers:**

**n → n/2 (n is even)**

**n → 3n + 1 (n is odd)**

**Using the rule above and starting with 13, we generate the following sequence:**

**13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1**

**It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.**

**Which starting number, under one thousand, produces the longest chain?**

collatz <- function(x) {

count <- 0

while (x > 1) {

if (x %% 2 == 0) {

x <- x / 2

count <- count + 1

}

else {

x <- 3 \* x + 1

count <- count + 1

}

}

return(count)

}

chain.len <- numeric(1e6)

for (i in 1:(1e6)) {

len.terms[i] <- collatz(i)

}

result <- which.max(chain.len)

cat("The result is:", result, "\n")

# for the figure presented here

steps <- numeric(1:1000)

starting.number <- 1:1000

for (i in starting.number) {

steps[i] <- collatz(i)

}

png(width = 600, height = 600)

plot(starting.number, steps, pch = 16,

main = "Collatz sequence with starting numbers no more than 1000")

dev.off()

**Q5. The number 512 is interesting because it is equal to the sum of its digits raised to some power: 5 + 1 + 2 = 8, and 83 = 512. Another example of a number with this property is 614656 = 284.**

**We shall define an to be the nth term of this sequence and insist that a number must contain at least two digits to have a sum.**

**You are given that a2 = 512 and a10 = 614656.**

**Find a30.**

List<BigInteger> a = new List<BigInteger>();

for (int b = 2; b < 400; b++) {

BigInteger value = b;

for (int e = 2; e < 50; e++) {

value \*= b;

if (DigitSum(value) == b) {

a.Add(value);

}

if (a.Count > 50) break;

}

if (a.Count > 50) break;

}