**Please, upload your answers in Canvas: assignment PC LAB – April 14**

Name: Vu The Doan

Student nr: 12918687

PC3a

**1**. An approximation for a factorial can be found using Stirling’s formula:

Write a function

stirling <- function(n)

and determine the ratio of this approximation and the true value:

n=1 n!/Stirling’s formula=1.08443755

n=2 n!/Stirling’s formula=1.04220712

…

n=10 n!/Stirling’s formula=1.00836536

Use

cat(sprintf("n=%2d n!/Stirling's formula=%.8f\n",...)

to show the ratio in 8 decimal places (you have to change the statement at ...).

R-code:

**stirling <- function(n) {**

**for(n in n:10) {**

**diff <- factorial(n)\*((sqrt(2\*pi\*n)\*(n/exp(1))^n))^(-1)**

**cat(sprintf("n=%2d n!/Stirling's formula=%.8f\n", n, diff))**

**}**

**}**

**2**. What will be the output of calling the function question()? Make use of memory tables below.

question<-function(){

a <- 0

b <- 1

c <- 2

d <- 3

a <- method1(b,c)

b <- method2(a,d)

cat("a=",a," b=",b,"c=",c," d=",d,"\n")

}

method1<-function(a,b){

c<-a/b

d<-a\*b

out<-method2(d,c)

return(out)

}

method2<-function(a,b){

return(a+b)

}

|  |  |
| --- | --- |
| **question()** | |
| a | 2.5 |
| b | 5.5 |
| c | 2 |
| d | 3 |
| **method1()** | |
| a | 1 |
| b | 2 |
| c | 0.5 |
| d | 2 |
| out | 2.5 |
| **method2()** | |
| a | 2 |
| b | 0.5 |
| **method2()** | |
| a | 2.5 |
| b | 3 |

**3**. and the sum of its digits is .

What is the sum of the digits of the number

The problem is that equals

1071508607186267320948425049060001810561404811705533607443750388370351\

0511249361224931983788156958581275946729175531468251871452856923140435\

9845775746985748039345677748242309854210746050623711418779541821530464\

7498358194126739876755916554394607706291457119647768654216766042983165\

2624386837205668069376,

which is “too large” to fit in standard integer R variable. Hence, we have to think of a way to make the computations exact. We are going to implement a multiplication algorithm ourselves. For instance, we can calculate as follows. First, we determine 2 times the last digit of , i.e. 6, which equals 12. Hence, the last digit of the multiplication equals 2. Since 12 is larger than 9, we set the carry to 1. The next digit of the outcome equals , where represents the carry. Since this is smaller than 10, we set the carry to 0. The third digit (from the right) equals . Again this is larger than 9, so we set the carry to 1. Since this is the last digit of and the carry equals 1, we must add a digit to the outcome that equals 1. Below are all the steps:

Step 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | 5 | 1 | 6 |  |  |
|  |  |  | 2 |  |  |
|  |  |  | 2 |  |  |

Step 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | +1 |  |  |
|  | 5 | 1 | 6 |  |
|  |  |  | 2 |  |
|  |  | 3 | 2 |  |

Step 3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 5 | 1 | 6 |  |
|  |  |  | 2 |  |
|  | 0 | 3 | 2 |  |

Step 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| +1 |  |  |  |  |
|  | 5 | 1 | 6 |  |
|  |  |  | 2 |  |
| 1 | 0 | 3 | 2 |  |

We will represent a number using a vector, where each digit is one element. So, x=c(5,1,6) represents then number 516.

Write a function called with the following header:

times2<-function (x)

It should take x as input argument and return a vector that represents x times 2. So,

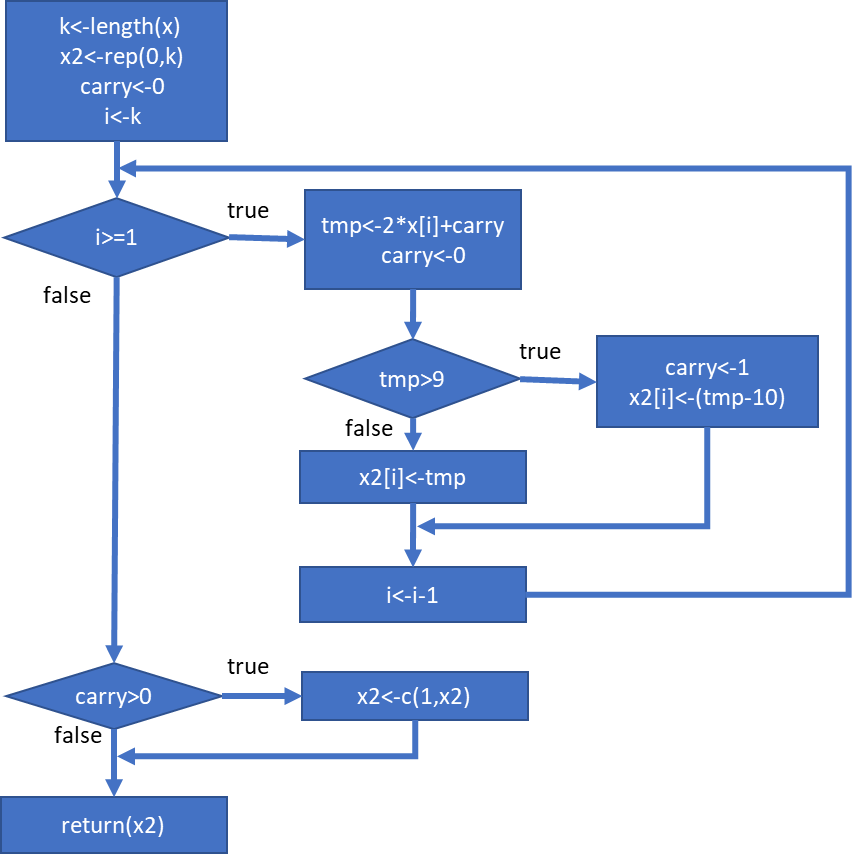
> times2(c(5,1,6))

[1] 1 0 3 2

Now, you can use

x<-times2(x)

within a for-loop that is executed 999 if you initialize x<-c(2). Below, you can find a flowchart of this function.



What is the sum of the digits of the number 2^1000?

**Answer**: 1366

**4**. Consider the following function:

The 2nd order approximation of in point *b* is given by:

The user chooses a value for the point *b*.

The *one* R-script has to

(i) calculate and its 2nd order approximation for *x* in [2.0,2.1,2.2,…,3.9,4.0];

(ii) determine in a *function*: **approx2 <- function(x,b)**

(iii) using **ggplot2**, plot the function and its approximation: and the approximation and add title and x-title; use dashed lines to join the markers that indicate the points

(iv) determine the normalized root mean squared error and show it in 7 decimals.

This is defined as:

where denotes the average of the values and *n* denotes the number of points. Please, use vectors only;

(v) show the ***first*** 6 values of *x* and the approximation in 5 decimals.

A close up of a map

Description automatically generated

**Enter value for b: 3**

**NRMSE for the 2nd order approximation is: 0.0227633**

**2.00000 2.10000 2.20000 2.30000 2.40000 2.50000**

**2.57293 2.49310 2.41711 2.34498 2.27668 2.21224**

Explanation: The user executes the red line at the command prompt (click on the Source button in Rstudio). The 3 (on the first line) is also entered by the user. The script shows the graph and blue text on the screen.

R-code:

b<-as.numeric(readline("Give a value for b:"))

f <- function(x){

exp(2/x)

}

g <- function(x,b) {

exp(2/b)-2\*(x-b)/(b^2)\*exp(2/b)+2\*(1+b)\*((x-b)^2)/(b^4)\*exp(2/b)

}

x <- seq(2,4,0.1)

cat("Values for f(x)\n")

print(f(x))

cat("Values for g(x)\n")

print(g(x,b))

table <- cbind(x, f(x), g(x,b))

colnames(table) <- c("x","f(x)", "g(x)")

table

table <- as.data.frame(table)

ggplot(table)

tablef <- data.frame(cbind(table[,1], table[,2]))

ggplot(tablef)