```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <Guano Anderson >


################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<-seq(10,1,-1)
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-seq(3,30,3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.

v3<-seq(1,291,10)
v4<-seq(2,292,10)
v5<-seq(3,293,10)
v6<-seq(4,294,10)
```

```r
v7<-seq(10,300,10)

M<-rbind(v3,v4,v5,v6,v7)
M
# 1.4 Print the first row of M
M[1,]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)

(log10(1)+log10(2))/((pi-1)/(pi-1))

log10(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-10^5
# 1.8 If you would now run the line
print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

## R is case sensitive,which means that u and V are two different variables

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

head(Eudata)
# 2.1 How many countries are there in the dataset?
nrow(Eudata)

# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
Eudata$Population[which.min(Eudata$Area)]# smallest

Eudata$Population[which.max(Eudata$Area)]#Largest

# 2.4 Calculate the number of countries that are members of the Eurozone
```

```
sum(Eudata$Eurozone==1)
# 2.5 Calculate the total GDP of all Eurozone members

sum(Eudata$GDP[Eudata$Eurozone==1])

# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU

#a
gdp_pc_Eu<-sum((Eudata$GDP)*1e6)/sum(Eudata$Population)
gdp_pc_Eu
#b

gdp_pc_EZ<-(sum(Eudata$GDP[Eudata$Eurozone==1])*1e6)/sum(Eudata$Population[Eudata$Eurozone==1])
gdp_pc_EZ
#c
gdp_pc_no_Eu_memb<-(sum(Eudata$GDP[Eudata$Eurozone==0])*1e6)/sum(Eudata$Population[Eudata$Eurozone==0])
gdp_pc_no_Eu_memb
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)
# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata<-subset(Eudata,Code != "GB")
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
One_roll_dice<-sample(1:6,1,T)
One_roll_dice
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,1000,T)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k==4 | k==5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<-sample(1:6,1000,T)
```

```r
z<-2*k-m
z
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock<-rnorm(100,0.12,0.2)
stock
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
mean(stock<0)

#     (b) by calculating the (theoretical) probability for a normal distribution
pnorm(0,mean = 0.12,sd=0.2)

##############################################
# 4: Functions and Optimization     (25 points)#
##############################################
# 4.1 Create the function f(x)=x^2 in R

f<-function(x){
  x^2
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f,from = -2,to=2)

# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
res<-optim(par=0,fn=f,method = "BFGS")
ximin<-res$par
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

grid_search<-optim(par = 0, fn= f, gr = 100, method = "L-BFGS-B", lower = -2, upper = 2)
xmin_grid<-
  optim()
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.
# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

checkLuhn<-function(v1){
  v1<-c(6,3,4,8,7)
  value<-length(v1)
  index<-1
  somma<-0
  while (value>=1) {
    if(index%%2==0){
      v1[value]<-v1[value]*2
      if(v1[value]>9){
        v1[value]<-v1[value]-9
      }
    }
    somma<-somma+v1[value]
    value<-value-1
    index<-index+1
  }
  if(somma%%10==0){
    return(TRUE)
  }else{
    return(FALSE)
  }

}
checkLuhn(digits)
```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <your name>
#Anna Bonera

################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- seq(10,1,-1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(3,30,3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.

# 1.4 Print the first row of M

# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1,1)
```

```r
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log(1)+log(2))/((pi+1)/(pi-1))
log((sin(2))/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u <- (10)^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

because the name of variables are case sensitive, so U is not the same as u
################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?

# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
sum(Eudata$Population[which.min(Eudata$Area)])
sum((Eudata$Population[which.max(Eudata$Area)]))
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone==TRUE])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
sum(Eudata$GDP/Eudata$Population)
sum(Eudata$GDP[Eudata$Eurozone==TRUE]/Eudata$Population[Eudata$Eurozone==TRUE])
sum(Eudata$GDP[Eudata$Eurozone==FALSE]/Eudata$Population[Eudata$Eurozone==FALSE])
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member

# 2.8 Calculate the number of EU founding members
EU_founding_members= sum(Eudata$Accession==1953)
```

```

# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"

# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



##############################################
# 3: Simulation and probability   (15 points)  #
##############################################
# 3.1 Use R to produce one roll of a dice.
dice=sample(1:6,1)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k <- sample(1:6,1000,replace = TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice

# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m

# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns

# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution


##############################################
# 4: Functions and Optimization     (25 points)#
##############################################
# 4.1 Create the function f(x)=x^2 in R
f <- function(x){x^2}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function

# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
```

```r

# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


y###################################################
# 5: Functions and algorithms    (25 points)       #
###################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))
```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                      <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                  #
################################################
# 0.1 Write your name below as a comment
# Spiatta Damiano


################################################
# 1: R basics    (13 points)                  #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<- 10:1
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(3,30, by = 3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M <- matrix(1:300, nrow = 10, ncol = 30)
head(M)
# 1.4 Print the first row of M
print(M[1,])
# 1.5 Display the last element of v1. Tell R to "display the last element
```

```r
of x",
#     regardless of the dimension of v1.
print(v2[length(v2)])
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7)^(1/5)
(log(1)+log(2))/((pi+1)/(pi-1))
log(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u <- 10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

#R in case sensitive, capital letters are not equals as normal letter. v1 is different than V1

###############################################
# 2: Data and Logical conditions  (20 points)  #
###############################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
sum()
# 2.2 Calculate the total population of the EU
sum(myData$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
myData$Population[which.min(myData$Area)]
myData$Population[which.max(myData$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(myData$IsEurozone == 1)
# 2.5 Calculate the total GDP of all Eurozone members
sum(myData$GDP[myData$IsEurozone == 1])
# 2.6 Calculate the GDP per capita in euros
#(a) of the total EU,
sum(myData$GDP) / sum(myData$Population)
#(b) of the Eurozone
sum(myData$GDP[myData$IsEurozone == 1]) /
  sum(myData$Population[myData$IsEurozone == 1])
#(c) of the non-Eurozone EU
sum(myData$GDP[myData$IsEurozone != 1]) /
  sum(myData$Population[myData$IsEurozone != 1])
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
myData$Accession == min(myData$Accession[myData$IsEurozone != 1])
# 2.8 Calculate the number of EU founding members
EU_founders <- sum(myData$Accession == 1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata[Eudata$CountryName != "United Kingdom", ]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata$Eurozone[Eudata$CountryName == "Bulgaria"] <- TRUE


###############################################
# 3: Simulation and probability   (15 points)  #
###############################################
# 3.1 Use R to produce one roll of a dice.
dice <- sample(1:6, size = 1, replace = TRUE)
dice
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
n <- 1000
k <- c(sample(1:6, size = n, replace = TRUE))
k
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
prob_4_or_5 <- mean(k %in% c(4, 5))
prob_4_or_5
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m <- sample(1:6, size = n, replace = TRUE)
z <- 2*k - m

expected_value_z <- mean(z)
expected_value_z
variance_z <- var(z)
variance_z
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock <- rnorm(100, mean = 0.12, sd = 0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
prob_neg_emp <- mean(stock<0)
print(prob_neg_emp)
#     (b) by calculating the (theoretical) probability for a normal distribution
prob_neg_theor <- pnorm(0,mean = 0.12, sd = 0.2)
```

```r
print(prob_neg_theor)

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f <- function(x){
  x^2
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
x <- seq(-2, 2, length.out = 200)
y <- f(x)
plot(x, y, type = "l", lwd = 2, main = "Plot of f(x) = x^2",xlab = "x", ylab = "f(x)")
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
opt <- optim(0,f,method = "BFGS")
x_min = opt$par
y_min = opt$value
xmin <- list("min x" = x_min, "min y" = y_min)
print(xmin)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
N <- 100
grid <- seq(-2, 2, length.out = N)
values <- f(grid)

x_min_grid <- grid[which.min(values)]
y_min_grid <- min(values)

xmin_grid <- list("min x" = x_min_grid, "min y" = y_min_grid)
print(xmin_grid)

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?

# The results are not identical because optim searches in a continuous space,
# while the grid search is limited to a finite set of discrete points.

```

```r

####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#       like this
#       a) process individual digits from right to left
#       b) leave digits number 1,3,5 etc (counted from right) unchanged
#       c) multiply digits 2,3,6 etc  (counted from right) by 2
#       d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#       e) calculate the sum of all (processed) digits
#       IF the result can be divided by 10, the number is a valid credit card number
#
#       Example: 63487
#       Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                      multiply 3 by 2 and do not subtract anything, do not change 6
#                      7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#       Hint: The operation x %% y yields the remainder of the division x/y.
#             For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

checkLuhn <- function(digits) {
  rev_digits <- rev(digits)

  for (i in seq_along(rev_digits)) {
    if (i %% 2 == 0) {
      rev_digits[i] <- rev_digits[i] * 2
      if (rev_digits[i] > 9) {
        rev_digits[i] <- rev_digits[i] - 9
      }
    }
  }

  sum(rev_digits) %% 10 == 0
}

#test
checkLuhn(10)
checkLuhn(36)
```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <your name>
#Eleonora Moroni

################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- seq(10,1, -1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(3,30,3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
row1 <- seq(1,291,10)
row2 <- seq(2,292,10)
row3 <- seq(3,293,10)
row4 <- seq(4,294,10)
row5 <- seq(10,300,10)
M <- rbind(row1,row2,row3,row4,row5)
# 1.4 Print the first row of M
```

```
M["row1",]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
x<-v1
tail(x,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log(1)+log(2))/((pi+1)/(pi-1))
log(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-(10)^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
#because u and U are seen in R as two diffrent variables, R is very senstivie

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
#28
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
Eudata$Population[min(Eudata$Area)]
Eudata$Population[max(Eudata$Area)]

# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==1)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone==1])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
#a)
sum(Eudata$GDP)/sum(Eudata$Population)
#b)
sum(Eudata$GDP[Eudata$IsEurozone==1])/sum(Eudata$Population[Eudata$
IsEurozone==1])
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)

# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata<-Eudata[Eudata$Country != "UK"]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
sample(1:6,1)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,1000, replace=TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
prob_4_or_5<- function(k)
{sum(k%in% c(4,5)/
       length(k))}
prob_4_or_5(k)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<-sample(1:6,1000,replace=TRUE)
z<-2*k-m
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock<-rnorm(100,0.12,0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
negative_stock<-sum(stock<0)/length(stock)
print(negative_stock)
################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
```

```
f<-function(x) {
  return<-x^2
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
x<-seq(-2,2)
plot(x,f(x), "l")
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin<-function(x){
  stopifnot(-2<=x && x<=2)
  return(x)
}

# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
```

```r
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))



```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                  #
################################################
# 0.1 Write your name below as a comment
# Erica Trofimov


################################################
# 1: R basics    (13 points)                  #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- seq (10, 1, -1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq( 3, 30, 3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
v1 = seq(1,291,10)
v2 = seq( 2, 292, 10)
v3 = seq( 3, 293, 10)
v4 = seq(4, 294, 10)
v5 = seq(10, 300, 10)

M = rbind(v1, v2, v3,v4,v5)
```

```r
# 1.4 Print the first row of M
M[1, ]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
v1[length(v1)]
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
## 1.
((12)/(19 -7 ))^(1 / 5)
## 2.
(log10(1) + log10(2))/ ((pi + 1)/ (pi - 1))

## 3.
log10 ( sin(2)/exp(2))

# 1.7 Create a variable "u" with the value "ten to the power minus 5"

u <- 10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
## R is case-sensitive. Hence, it would consider U as a totally different object than U. We do not have any U object
## in our directory, hence R will give error. If we want to print the value of the variable we just created,
## we would need to run print(u).

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location

Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
min_pop = which.min(Eudata$Area)
max_pop = which.max(Eudata$Area)
Eudata$Population[min_pop]
Eudata$Population[max_pop]
```

```

# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone)
# 2.5 Calculate the total GDP of all Eurozone members
library(dplyr)
Eurozone%>%
  filter ( Eurozone == 1) %>%
  summarize( sum(GDP))
 ## or
Eurozone = Eudata[ Eudata$Eurozone == 1,  ]
sum(Eurozone$GDP)

# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
a = sum(Eudata$GDP)/sum(Eudata$Population)
b = sum(Eudata$GDP)/sum(Eurozone$Population)
NonEuro = Eudata[ Eudata$Eurozone == 0,  ]
c = sum(Eudata$GDP)/sum(NonEuro$Population)


# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member

found_year = Eudata$Accession[which.min(Eudata$Accession) ]

# 2.8 Calculate the number of EU founding members
nrow( Eudata[ Eudata$Accession ==  1953, ] )
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata[ Eudata$Code == 'UK', ]
Eudata[ - 28, ]

# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata[ Eudata$CountyName == 'Bulgaria', 'Eurozone'] <- 1


################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
sample(1 : 6, 1, replace  = TRUE)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k <- sample(1:6, 1000, replace = TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
k4 = sum(k ==4)
k5 = sum(k == 5)
prob_k4_5 = (k4 + k5 ) / (length(k))
```

```r

# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m = sample(1:6, 1000, replace = TRUE)

expr <- function(k,m){
  return(2*k - m )
   }
z = expr ( k, m)
z_mean = mean(z)
z_var = var(z)

# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock <- rnorm(100, 0.12, 0.3)


# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
neg_stock = sum( stock < 0)
prob_neg = neg_stock / length(stock)
prob_neg
#     (b) by calculating the (theoretical) probability for a normal distribution
pnorm(  0 , neg_stock ,  lower.tail = TRUE)

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f = function(x){
  return ( x^2)
  }
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
plot( f, xlim = c(-2,2))
#     If in doubt, type "?plot" to get the help file for the function

# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
opt = optim(
  par = 2,
  f,
  method = 'BFGS'
  )
opt$par
xmin = opt$value
f(xmin)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
N = 100
points = seq( -2, 2 , 0.04)
y_grid = sapply( points, f)
min_index = which.min(y_grid)
best_y_grid_search = y_grid[ min_index]
xmin_grid = x_grid [ min_index]

min_coordinates = c(xmin_grid, best_y_grid_search )
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?
## Results of 4.4 depend on the arbitrary parameter we want the optim() function
## to start iterating from. Based on that, results slightly vary. Moreover, it also depends on the algorithm we chose.
## Since the result of this minimization problem is 0, chosing a Gradient descent or a Nelder-Mead cuould
## give slightly different results (but still right if we approximate)

## The grid search, instead, is a brute force method. It is still effective,
## but highly depends on the value we decide to look from. If 0 was not in the grid, and we would have added, say, 0.1 as the value
## closer to 0, then this method would have gave us 0.1 as xmin value, completely ignoring 0 which was not in our grid.

y#####################################################
# 5: Functions and algorithms    (25 points)       #
#####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left

#      b) leave digits number 1,3,5 etc (counted from right) unchanged

#      c) multiply digits 2,3,6 etc  (counted from right) by 2


#      d) if a digit (after multiplying by 2) is larger than 9, subtract
```

```
9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

checkLuhn <- function(digits) {
    right_digits <- rev(digits)

    for (i in 1:length(right_digits)) {
      if (i %% 2 == 0) {
        doubled <- right_digits[i] * 2
        if (doubled > 9) {
         doubled = doubled - 9
        }
        right_digits[i] <- doubled
      }
    }
    sum <- sum(right_digits)

    if (sum %% 10 == 0) {
      return(TRUE)
    } else {
      return(FALSE)
    }
  result <- checkLuhn(card_to_check)
  }
# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.


# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))
checkLuhn(digits )

```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Filippo Manachino    (2 points)                 #
################################################
# 0.1 Write your name below as a comment
# <Filippo Manachino>


################################################
# 1: R basics    (13 points)                 #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- 10:1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(3, 30, by=3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M <- matrix(1:300, nrow=10, ncol=30)
# 1.4 Print the first row of M
M[1, ]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
v1[length(v1)]
```

```r
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
((sqrt^5(12/(19-17))))
(log(1)+log(2))/((pi+1)/(pi-1))
log((sin(2))/(exp(1))^2)
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u <- 10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
#R is case-sensitive: U and u are different objects (different names)

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata) #this is just the command, to save this into a variable just do n_countries <- nrow(Eudata)
# 2.2 Calculate the total population of the EU
total_pop <- sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
pop_smallest_by_area <- Eudata$Population[which.min(Eudata$Area)]
pop_biggest_by_area <- Eudata$Population[which.max(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
n_eurozone <- sum(Eudata$Eurozone==1)
# 2.5 Calculate the total GDP of all Eurozone members
gdp_eurozone <- sum(Eudata$GDP[Eudata$Eurozone==1])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
total_gdp <- sum(Eudata$GDP)
gdp_eurozone <- sum(Eudata$GDP[Eudata$Eurozone==1])
gdp_non_eurozone <- sum(Eudata$GDP[Eudata$Eurozone==0])

total_gdp_pc <- total_gdp/sum(Eudata$Population)
gdp_pc_eu <- gdp_eurozone/sum(Eudata$Population[Eudata$Eurozone==1])
gpd_pc_non_eu <- gdp_non_eurozone/sum(Eudata$Population[Eudata$Eurozone==0])
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
```

```r
eu_founding <- min(Eudata$Accession)
# 2.8 Calculate the number of EU founding members
total_founders <- sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata <- subset(Eudata, Code != "UK")
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.


#################################################
# 3: Simulation and probability   (15 points)  #
#################################################
# 3.1 Use R to produce one roll of a dice.
dice <- sample(1:6, 1)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k <- sample(1:6, 1000, replace = TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
p4_p5 <- mean(k==4|k==5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m <- sample(1:6, 1000, replace = TRUE)
z <- 2*k-m
Expected <- mean(z)
Variance <- var(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock <- rnorm(n = 100,mean = 0.12, sd = 0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
p_neg_hat <- mean(stock<0)
#     (b) by calculating the (theoretical) probability for a normal distribution
p_neg_theory <- pnorm(0, mean = 0.12, sd = 0.2)

#################################################
# 4: Functions and Optimization     (25 points)#
#################################################
# 4.1 Create the function f(x)=x^2 in R
f <- function(x){x^2}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
x <- seq(-2,2,length.out=100)
```

```r
plot(x,f(x),type="l")
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin <- optim(0, f, method = "L-BFGS-B",lower=-2, upper=2)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
x <- seq(-2,2,length.out=100)
xmin_grid <- grid()
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


y####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2, the sequence in ambigous, i used 2,4,6 etc
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
#the sequence in ambigous, i used 2,4,6 etc
digits <- as.numeric(unlist(strsplit("5019717010103742","")))
d <- c(2,4,7,3,0,1,0,1,0,7,1,7,9,1,0,5)
checkLuhn <- function(d){digits}
for (i in 1:16){
  if (d=="1,3,5,7,9"){
    creditnumber <- d
  }
  else if(d=="2,4,6,8"){
    h <- d*2
    if(h>9){
      h <- h-9
      creditnumber <- h
    }
  }
}
#unfortunately i dont remember how to code it, the base idea of my algorithm was to create a for cycle that would scan the vector number by number using an if condition
#if the number was 1,3,5,7,9 it would add it to a vector called creditnumber, else if the number was 2,4,6,8 it would first multiply it by 2 then with the other if it would check
#if the number was >9 in that case it would divide by 2 and then add it to the vector
#at the end outside the vector i would have summed it and checked if the division by 10 would leave an integer or no so to determine if the credit number was valid or no


```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <your name>

##### <Leonardo Zanga>
################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<-seq(10,1,-1)
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-3*seq(1:10)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M<-rbind(seq(1,291,10),seq(2,292,10),seq(3,293,10),seq(4,294,10),seq(10,300,10))
M
# 1.4 Print the first row of M
M[1,]
```

```r
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
M[length(M)]
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^1/5
(log(1)+log(2))/((pi+1)/(pi+2))
log(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
 #### Because U and u are different values in R


################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)

# 2.2 Calculate the total population of the EU
sum(Eudata$Population[Eudata$Eurozone==1])
# 2.3 Print the population of the smallest and largest EU country by Area
Eudata$Population[which.max(Eudata$Area)]
Eudata$Population[which.min(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
Sum(Eudata$Eurozone==TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone])

# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
sum(Eudata$GDP/Eudata$Population)
sum(Eudata$GDP[Eudata$Eurozone==TRUE]/Eudata$Population[Eudata$Eurozone==TRUE])
sum(Eudata$GDP[Eudata$Eurozone==FALSE]/Eudata$Population[Eudata$Eurozone==FALSE])
```

```
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accesion)

# 2.8 Calculate the number of EU founding members
sum(Eudata$CountyName[Eudata$Accesion==1953])
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata<-Eudata[Eudata$countyname=='United Kingdom',]
Eudata

# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
new<-bulgaria
Eudata$New


################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
R<-sample(1:6,1,replace=TRUE)
R
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,100,replace=TRUE)
k
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
k[k>=4 & k<=5]
length(k[k>=4 & k<=5])/length(k)

# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<-sample(1:6,1000,replace=TRUE)
m
z=2*k-m
mean(z)

# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stocks<-rnorm(100,0.12,0.2)
stocks
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
prob<-mean(stocks<0)
```

```
prob
#     (b) by calculating the (theoretical) probability for a normal distribution
pnorm(0,0.12,0.2)

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f<-function(x){
  ris<-x^2
  print(ris)
}
# 4.2 Calculate the value of f for x=1
f(1)

# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
plot(f,2,-2)
plot()
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin<-optim( par=0, fn=f, method="BFGS")
xmin
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
lowerbound=-2
upperbound=+2
xmin_grid<-optim( par=100, fn=f, method="BFGS",lower=lowerbound, upper=upperbound)
xmin
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?
because i have add upper and lower bound


y####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.
s=0
v1<-c(6,3,4,8,7)
n<-v1()

chwcluLuhn<-function(v1){
  if(n*2>9){}
  ris< n+((n+1)*2-9)+4+3*2+6/10
  valid<-(ris/10)==TRUE
  s=s+s1
}
  else{

}
# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))



```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# Dell Erba Lisa


################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<-10:1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-seq(3,30,by=3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
row1<-seq(1,291,by=10)
row2<-seq(2,292,by=10)
row3<-seq(3,293,by=10)
row4<-seq(4,292,by=10)
row5<-seq(10,300,by=10)
M<-rbind(row1,row2,row3,row4,row5)
# 1.4 Print the first row of M
```

```r
M[1,]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log10(1)+log10(2))/((pi+1)/(pi-1))
log10(sin(2)/exp(2))

# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
#R is a case sensitive, our variable is called u and not U, which is undefined.

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)
# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
tot_pop<-sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
min(Eudata$Population)
max(Eudata$Population)
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==1)
# 2.5 Calculate the total GDP of all Eurozone members
gdp_eurozone<-sum(Eudata$GDP[Eudata$Eurozone==1])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
gdp_tot<-sum(Eudata$GDP)
gdp_per_capita_eu<-gdp_tot/tot_pop
pop_eurozone<-sum(Eudata$Population[Eudata$Eurozone==1])
gdp_per_capita_eurozone<-gdp_eurozone/pop_eurozone
gdp_non_eurozone<-sum(Eudata$GDP[Eudata$Eurozone==0])
pop_non_eurozone<-sum(Eudata$Population[Eudata$Eurozone==0])
```

```r
gdp_per_capita_non_eurozone<-gdp_non_eurozone/pop_non_eurozone
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)
# 2.8 Calculate the number of EU founding members
founding_memebers<-sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata<-Eudata[Eudata$CountyName!="United Kingdom", ]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata<-Eudata[Eudata$CountyName=="Bulgaria", ]


################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
dice<-sample(1:6,1,replace=TRUE)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,1000,replace=TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k==4 | k==5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<-sample(1:6,1000,replace=TRUE)
z<-2*k-m
expected_value_z<-mean(z)
print(expected_value_z)
var(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock<-rnorm(100,0.12,0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution

#a
mean(stock<0)
#b
stock<-pnorm(0,0.12,0.2)
################################################
# 4: Functions and Optimization     (25 points)#
```

```r
################################################
# 4.1 Create the function f(x)=x^2 in R
f<-function(x){
  x<-x^2
  return(x)
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f, -2,2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin<-optim(par=0,fn=f,lower=-2,upper=2,method='L-BFGS')
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
xmin<-optim(par=100,fn=f,method='Brent', lower=-2,upper=2)

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?
# Because of the floating-point approximation
#The two ways to find the location of the minimum are close to each other but they aren't exactly the same
y####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

checkLuhn<-function(CardNumber){
  digits<-as.numeric(unlist(strsplit(CardNumber, "")))
digits_rev<-rev(digits)
for(i in seq_along(digits_rev)){
  if( i %% 2==0){
    digits_rev[i]<-digits-rev[i]*2
    if(digits_rev[i]>9){
      digits_rev[i]<-digits_rev[i]-9
    }
  }

}
total<-sum(digits_rev)
return(total%%10==0)
}

checkLuhn("5019717010103742")

```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


###############################################
# 0: Your name    (2 points)                  #
###############################################
# 0.1 Write your name below as a comment
# <your name>
#Martinelli Ludovico

###############################################
# 1: R basics    (13 points)                  #
###############################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<- seq(10,1,-1)
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-seq(3,30,3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
riga1<- seq(1,291,10)
riga2<-seq(2,292,10)
riga3<-seq(3,293,10)
riga4<-seq(4,294,10)
riga10<-seq(10,300,10)
```

```r
M<-matrix(c(riga1,riga2,riga3,riga4,riga10), nrow = 5, byrow = T)
M
# 1.4 Print the first row of M
M[1,]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log10(1)+log10(2))/((pi+1)/(pi-1))
log10(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-10^12
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

#R is case sensitive, so we have to pay attention when we write capital letter.

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)
View(Eudata)
# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
Eudata$Population[which.min(Eudata$Area)]
Eudata$Population[which.max(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==1)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone==1])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
gdp_per_capita<-(Eudata$GDP/Eudata$Population)
gdp_per_capita
```

```r
sum(gdp_per_capita[Eudata$Currency=='euro'])
sum(Eudata$GDP[Eudata$Eurozone==1]/Eudata$Population[Eudata$Eurozone==1])
sum(Eudata$GDP[Eudata$Eurozone==0]/Eudata$Population[Eudata$Eurozone==0])

# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
Eudata$Accession[which.min(Eudata$Accession)]

# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata<-Eudata[Eudata$CountyName != 'United Kingdom',]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata$Accession[3,4]<- 2026
#qui devo cercare di sostituire la casella accession nel rigo 3, mettendo la data 2026


################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
sample(1:6,1)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,1000, replace = TRUE)
k
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
prob4<-mean(k==4)
prob4
prob5<-mean(k==5)
prob5
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<- sample(1:6,1000, replace = T)
m
k
z<-2*k-m
expectedvalue<-mean(z)
round(expectedvalue)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock<-rnorm(mean=0.12,sd=0.2,100)
stock
```

```
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(pnorm(stock<0))
round(mean(pnorm(stock<0)))
################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f<- function(x){
  x^2
}

# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
plot(f,-2,2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin<- optim(par=1,fn=f,gr=NULL, method = 'BFGS')
xmin
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

#in questo caso ho provato ad utilizzare questo metodo dell'ottimizzazione con grid
#nonostante non riesca a svolgerlo
kgrid<-seq(0,100,1)
ugrid<-c(kgrid)
k_ottimo<- kgrid[which.min(ugrid)]
k_ottimo
xmin_grid<- optim(par=1,fn=kgrid, gr=grad ,lower=-2, upper=2 ,method = 'BFGS')

#in questo caso devo utilizzare il comando grid research per trovare il
#punto di minimo della variabile xmin_grid
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?
#4.4 is negative infinity while the 4.5 is between -2 and 2

```

```
#####################################################
# 5: Functions and algorithms    (25 points)       #
#####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.
checkLuhn<- function(x){
  count= 0
  if(x==1 && 3 && 5){
    return(x)
  }
  else if(x==2 && 3 && 6){
    return(x*2)
  }
  else if (x*2>9){
    return(x-9)
  }
  return(x)
}

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


###############################################
# 0: Your name    (2 points)                  #
###############################################
# 0.1 Write your name below as a comment
# <Malak El fatih>


###############################################
# 1: R basics    (13 points)                  #
###############################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 = seq(10,0,-1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 = 3 *(1:10)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M <- matrix(seq(1, 300, by = 1), nrow = 5, byrow = TRUE)
# 1.4 Print the first row of M
M[1, ]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.

```

```r
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
res <- c((12/(19-7))^(1/5),
         (log(1) + log(2)) / ((pi + 1)/(pi - 1 )),
         log(sin(2)/exp(2))
)
# 1.7 Create a variable "u" with the value "ten to the power minus 5"

# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.


################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)

# 2.2 Calculate the total population of the EU
sum(Eudata$Population)

# 2.3 Print the population of the smallest and largest EU country by Area

pop_smallest <- Eudata$Population[which.min(Eudata$Area)]

pop_largest <- Eudata$Population[which.max(Eudata$Area)]

pop_smallest
pop_largest

# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$IsEurozone == 1)

# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$IsEurozone == 1])

# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
# (a)
```

```r
gdp_pc_eu <- sum(Eudata$GDP) * 1e6 / sum(Eudata$Population)

# (b)
gdp_pc_euro <- sum(Eudata$GDP[Eudata$IsEurozone == 1]) * 1e6 /
  sum(Eudata$Population[Eudata$IsEurozone == 1])

# (c)
gdp_pc_noneuro <- sum(Eudata$GDP[Eudata$IsEurozone == 0]) * 1e6 /
  sum(Eudata$Population[Eudata$IsEurozone == 0])

gdp_pc_eu
gdp_pc_euro
gdp_pc_noneuro

# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member

# 2.8 Calculate the number of EU founding members

# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"

# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
sample(1:6, 1)

# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k <- sample(1:6, 1000, replace = TRUE)

# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k == 4 | k == 5)

# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m <- sample(1:6, 1000, replace = TRUE)

# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
```

```r
stock <- rnorm(100, mean = 0.12, sd = 0.2)

# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(stock < 0)
pnorm(0, mean = 0.12, sd = 0.2)


################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f <- function(x) {
  x^2
}

# 4.2 Calculate the value of f for x=1
f(1)
# [1] 1


# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
x <- seq(-2, 2, length.out = 400)
plot(x, f(x), type = "l", xlab = "x", ylab = "f(x)", main = "f(x) = x^2 on [-2, 2]")


# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin <- optimize(g, interval = c(-10,10))$minimum

# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
N <- 100
grid <- seq(-2, 2, length.out = N)
vals <- f(grid)

xmin_grid <- grid[which.min(vals)]
xmin_grid

```

```
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?




y##################################################
# 5: Functions and algorithms    (25 points)       #
##################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
rev_digits <- rev(digits)
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

checkLuhn <- function(digits) {
  digits <- as.integer(digits)
  if (any(is.na(digits)) || any(digits < 0 | digits > 9)) return(FALSE)

  a) process individual digits from right to left
```

```
rev_digits <- rev(digits)

#leave digts number 1,3,5 ecc.. (counted from right) unchanged
pos <- seq_along(rev_digits)

# multiply digits 2,3,6 ecc..  (counted from rigt) by 2
to_double <- (pos %% 2 == 0)
rev_digits[to_double] <- rev_digits[to_double] * 2

#if a digit (after multiplying by 2) is larger than 9, subtract 9
rev_digits[rev_digits > 9] <- rev_digits[rev_digits > 9] - 9

# calculate the sum of all (processed) digits
sum(rev_digits) %% 10 == 0
```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <Matteo Gangi>
# "Matteo Gangi


################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1=seq(10, 1, -1)

# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2= seq(1, 30, 3)

# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M= rbind(seq(1, 291,10), seq(2, 292, 10), seq(3, 293, 10), seq(4,294,10), seq(10,300, 10))

# 1.4 Print the first row of M
```

```
M[1,]

# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1, 1)

# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7)) ^ (1/5)
(log10(1) + log0(2))/ ((pi+1)/(pi-1))
log10(sin(2)/exp(2))

# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u= 10^(-5)

# 1.8 If you would now run the line
# print(U)

# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

# Because R is case sensitive


################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)

# 2.2 Calculate the total population of the EU
sum(Eudata$Popolation)


# 2.3 Print the population of the smallest and largest EU country by Area
smallest= Eudata$Population[which.min(Eudata$Area)]
largest = Eudata$Population [which.max(Eudata$Area)]

# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone== TRUE)

```

```
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone== TRUE])

# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU

sum(Eudata$GDP/Eudata$Population)                #a
sum(Eudata$GDP[Eudata$Eurozone== TRUE]/Eudata$Population[Eudata$Eurozone== TRUE])       #b
sum(Eudata$GDP[Eudata$Eurozone== FALSE]/Eudata$Population[Eudata$Eurozone== FALSE])

# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)

# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession == 1953)

# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata= Eudata[Eudata$CountyName != "United Kingdom"]


# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)   #
################################################
# 3.1 Use R to produce one roll of a dice.
dice= sample(1:6, 1, TRUE)

# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k=sample (1:6, 1, TRUE)
k=sample (1:6, 1, TRUE)

# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean (k==4 | k==5)

# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m

m= sample(1:6, 1000, TRUE)
```

```
z= 2*K - m
mean(z)

# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock = rnorm(100, 0.12, 0.2)

# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(stock<0)      #a
pnorm(0, 0.12, 0.2)       #b

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f = function(x){
  x^2
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f, -2, 2)

# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
neg_f = function (x){
  -f(x)
}
res= optim(par=0, fn =neg_f, method = "Brent", lower= -2, upper=2)
xmin= res$par
print(xmin)

# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid


# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4)
```

```
and (4.5)
#     are not identical. Why?


y####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))



```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <your name>
<Matteo Zucchi>

################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- 10:1
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(3, 30, 3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M= rbind(seq(1, 291, 10), seq(2, 292, 10), seq(3, 293, 10), seq(4, 294, 10), seq(10, 300, 10))
# 1.4 Print the first row of M
M[1,]
# 1.5 Display the last element of v1. Tell R to "display the last element
```

```
of x",
#     regardless of the dimension of v1.
tail(v1,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log10(1)+log10(2))/((pi+1)/(pi-1))
log10(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u= 10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
--> Because R is case sensitive

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum=(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
smallest= Eudata$Population[which.min(Eudata$Area)]
largest= Eudata$Population[which.max(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone == TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone == TRUE])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
sum(Eudata$GDP/Eudata$Population) # (a)
sum(Eudata$GDP[Eudata$Eurozone == TRUE]/Eudata$Population[Eudata$Eurozone == TRUE]) #(b)
sum(Eudata$GDP[Eudata$Eurozone == FALSE]/Eudata$Population[Eudata$Eurozone == FALSE]) # (c)
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)
```

```
# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession == 1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata= Eudata[-28,]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata= Eudata[Eudata$CountyName ]


################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
dice= sample(1:6, 1, TRUE)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k= sample(1:6, 1000, TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k == 4 | k == 5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m= sample(1:6, 1000, TRUE)
z= 2*k-m
mean(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock= rnorm(100, 0.12, 0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(stock<0) #(a)
pnorm(0, 0.12, 0.2) #(b)

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f= function(x){
  x^2
}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
```

```
curve(f, -2, 2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
neg_f= function(x){
  -f(x)
}
res= optim(par=0, fn = neg_f,method = "Brent", lower = -2, upper = 2)
xmin= res$par
print(xmin)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
grid()
xmin_grid
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


y####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

```

```r
# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

checkLuhn <- function(x){
  counter<- 0
  digits<-
}


```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <your name>
#<Nicholas Serantoni>

################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<- (10:1)
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-seq(3,30,by=3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
row1<- seq(1,291, by=10)
row2<-seq(2,292,by=10)
row3<- seq(3,293,by=10)
row4<-seq(4,294, by=10)
row5<- seq(10,300, by=10)
```

```
M<-rbind(row1,row2,row3,row4, row5)
M
# 1.4 Print the first row of M
M["row1",]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
v1<-(10:1)
v1
tail(v1,1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12/(19-7))^(1/5)
(log10(1)+log10(2))/((pi+1)/(pi-1))
log10(sin(2)/exp(2))
gx<- function(x){
  g<-3*X^2-x/2+2
  g
}
gx
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-10^5
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
#1.8"u"and "U" are seen as two different variables.we should use the same lower case name for variables

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)
Eudata<- read.csv("~/Desktop/Inf2mock2024 (2)/Eudata.csv")
# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
min(Eudata$Area)
max(Eudata$Area)
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone)
# 2.5 Calculate the total GDP of all Eurozone members
```

```
sum(Eudata$GDP[myData$Eurozone])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
percepita<-Eudata$GDP/Eudata$Population
#a)
sum(percepita)
#b)
sum(percepita(Eudata$Eurozone))
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
EUfoundation<-min(Eudata$Accession)
EUfoundation
# 2.8 Calculate the number of EU founding members
EUfoundation<-sum(Eudata)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata=Eudata[-28]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
sample(1:6,1, replace=T)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-sample(1:6,10000,replace =T)
k
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k==4 | k==5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m=sample(1:6,10000,TRUE)
z=2*k-m
mean(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock<- rnorm(100,mean=0.12,sd=0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
empirical_probability<-sum(stock<0)/length(stock)
```

```r
#     (b) by calculating the (theoretical) probability for a normal distribution
theoretical_probability<-pnorm(0,mean=0.12,sd=0.2)

################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
fx<- function(x){
  fx1<- -x^2+1
  print(fx1)
}
fx
# 4.2 Calculate the value of f for x=1
fx(x=1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
plot(fx,-2,2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
 xmin<-optimize(fx, interval=c(-2,2))$minimum
 print(xmin)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


y####################################################
# 5: Functions and algorithms    (25 points)        #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
```

```r
#
#       Example: 63487
#       Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                      multiply 3 by 2 and do not subtract anything, do not change 6
#                      7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#       Hint: The operation x %% y yields the remainder of the division x/y.
#             For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.
checkLuhn<- function
# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))



```

```
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


###############################################
# 0: Your name    (2 points)                  #
###############################################
# 0.1 Write your name below as a comment
# <your name>

# Samuel Boccomino

###############################################
# 1: R basics    (13 points)                  #
###############################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 <- seq(from = 10, to = 1, by = -1)
v1
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 <- seq(from = 3, to = 30, by = 3)
v2
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
r1 <- seq(from = 1, to = 291, by = 10)
r2 <- seq(from = 2, to = 292, by = 10)
r3 <- seq(from = 3, to = 293, by = 10)
r4 <- seq(from = 4, to = 294, by = 10)
```

```
r5 <- seq(from = 10, to = 300, by = 10)
M <- rbind(r1, r2, r3, r4, r5)
M
# 1.4 Print the first row of M
M[1 ,]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
tail(v1, 1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
((12/(19-7)))^(1/5)
(log(1)+log(2))/((pi+1)/(pi-1))
log(sin(2)/(exp(1))^2)
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u <- 10^(-5)
u
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

# It doesn't work because R is case sensitive, so U is different from u.

################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata-2.csv', sep=";",header=TRUE)
# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
Eudata$Population[which.min(Eudata$Area)]
Eudata$Population[which.max(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone==TRUE])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
# (a)
total_GDP_euros <- sum(Eudata$GDP)*10^6
```

```
total_population <- sum(Eudata$Population)
GDP_per_capita_total <- total_GDP_euros / total_population
GDP_per_capita_total
# (b)
GDP_eurozone <- sum(Eudata$GDP[Eudata$Eurozone==TRUE])*10^6
pop_eurozone <- sum(Eudata$Population[Eudata$Eurozone==TRUE])
GDP_per_capita_eurozone <- GDP_eurozone/ pop_eurozone
GDP_per_capita_eurozone
# (c)
GDP_non_euro <- sum(Eudata$GDP[Eudata$Eurozone==FALSE])*10^6
pop_non_euro <- sum(Eudata$Population[Eudata$Eurozone==FALSE])
GDP_per_capita_non_euro <- GDP_non_euro / pop_non_euro
GDP_per_capita_non_euro
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession)
# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession==1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata <- Eudata[Eudata$CountyName!="United Kingdom", ]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.

################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
dice_roll <- sample(1:6,1)
dice_roll
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k <- sample(1:6,1000,replace = TRUE)
k
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
p4 <- k[k==4]
p5 <- k[k==5]
probability <- (length(p4)+length(p5))/1000
probability
percentage <- ((length(p4)+length(p5))/1000)*100
percentage
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m <- sample(1:6,1000,replace = TRUE)
m
z <- 2*k-m
z
mean(z)
```

```r
var(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
stock <- rnorm(100,mean=0.12,sd=0.2)
stock
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution

# (a)
negative_stock_return1 <- sum(stock<0)/length(stock)
negative_stock_return1
# (b)
negative_stock_return2 <- pnorm(0,mean=0.12,sd=0.2)
negative_stock_return2
################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f <- function(x){x^2}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f,-2,2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin <- optim(-2, f, method = 'L-BFGS-B', lower = -2, upper = 2)
xmin
# just the par:
xmin$par
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
grid <- seq(from = -2, to = 2, by = 0.04)
best_value <- Inf
best_x <- NA
for(x in grid){
  current_value <- f(x)
  if(current_value < best_value){
    best_value <- current_value
    best_x <- x
  }
}
xmin_grid <- print(best_value)
cat("Minimum of the function:", best_value, "\n")
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?

# Because f(x) is a continuos function

####################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

checkLuhn <- function(credit_card_number){
  digits <- as.numeric(unlist(strsplit(as.character(credit-card_number))))
  n <- lenght(digits)
  for(i in seq(n, 1, -2)){
    if(i>1){
      digits[i-1] <- digits[i-1]*2
      if(digits[i-1]>9)
        digits[i-1] <- digits[i-1]-9
    }
  }
  return(sum(digits)%%10==0)
}
# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))
digits


```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


###############################################
# 0: Your name   (2 points)
name<-StefanoLaureti
###############################################
# 0.1 Write your name below as a comment
# <your name>
# <Stefano Laureti>

###############################################
# 1: R basics    (13 points)                  #
###############################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1<-seq(10,1,-1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2<-seq(3,30, 3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M <- matrix(seq(1, 300, 2), nrow=5, byrow=FALSE)
# 1.4 Print the first row of M
M[1, ]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
```

```r
tail(v1, 1)

# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
calc_1_7 <- c(
  (12/(19-7))^(1/5),
  (log10(1) + log10(2)) / ((pi + 1)/(pi - 1)),
  log10(sin(2) / exp(2))
)
calc_1_7


# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u<-c(10^-5)
u

# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.
# R is case-sensitive: U and u are different objects; also Print() is not print().


################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population, na.rm=TRUE)
# 2.3 Print the population of the smallest and largest EU country by Area
pop_smallest <- Eudata$Population[which.min(Eudata$Area)]
pop_largest  <- Eudata$Population[which.max(Eudata$Area)]
pop_smallest
pop_largest

# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$IsEurozone == 1, na.rm = TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$IsEurozone == 1], na.rm = TRUE)
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU
gdp_pc_all <- (sum(Eudata$GDP, na.rm = TRUE) * 1e6) / sum(Eudata$Population, na.rm = TRUE)

gdp_pc_ez <- (sum(Eudata$GDP[Eudata$IsEurozone == 1], na.rm = TRUE) * 1e6) /
  sum(Eudata$Population[Eudata$IsEurozone == 1], na.rm = TRUE)

gdp_pc_nonez <- (sum(Eudata$GDP[Eudata$IsEurozone == 0], na.rm = TRUE) * 1e6) /
  sum(Eudata$Population[Eudata$IsEurozone == 0], na.rm = TRUE)

gdp_pc_all
gdp_pc_ez
gdp_pc_nonez
# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
min(Eudata$Accession, na.rm = TRUE)
# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession == 1953, na.rm = TRUE)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata <- Eudata[Eudata$`County Name` != "United Kingdom", ]
# (se il nome è diverso, controllo con: names(Eudata))
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.



################################################
# 3: Simulation and probability   (15 points)  #
################################################
# 3.1 Use R to produce one roll of a dice.
rolldice<-sample(1:6,1,replace=TRUE)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k<-seq(sample(1:6,5000,replace=TRUE))
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
prob<- sum(k%%4)
prob_4<-prob/length(k)
prob2<- sum(k%%5)
prob_5<-prob2/length(k)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m<-seq(sample(1:6,1000,replace=TRUE))
z<-c(2*k-m)
z
```

```
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
mu<-0.12
sdr=sqrt(0.2)
stock100<-rnorm(100, mean=mu, sd=sdr)
sum(stock100>0)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
pnorm(0, mean=mu, sd=sdr)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(pnorn(0, mean=mu, sd=sdr)<0)


################################################
# 4: Functions and Optimization     (25 points)#
################################################
# 4.1 Create the function f(x)=x^2 in R
f<-function(x) x^2
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f, from = -2, to = 2)
# 4.4 Numerically, by using R, find the location of the minimum using the optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
xmin <- optimize(f, interval = c(-2, 2))$minimum
xmin
f(xmin)
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid
xmin_grid <- optimize(f, interval = c(-2, 2), n=100)$minimum
xmin_grid
f(xmin_grid)
# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?

# Because x^2 -> -Inf as |x| -> Inf, so the function is unbounded below.
####################################################
# 5: Functions and algorithms    (25 points)       #
```

```
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
# if (n %% 2 > 9) { n <- n-9 }


Luhn<-function(n) { if (x>=0 & x<9){

}

}

  if (n %% 2 > 9) { n <- n-9
  }
  while (n != 1) {
    n <- if (n %% 2 == 0) n/2 else 3*n + 1
    out <- c(out, n)
  }
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))

```

```r
# Exam "Informatica 2", 2025-09-04
# Peter Gruber and Paul Schneider

# Rules
# - Except for theoretical questions, all questions must be answered using R!
# - The exam takes 90 minutes and has 5 questions
# - There are a total of 100 points
# >>>>> SUBMIT your answers on iCorsi in time                       <--- IMPORTANT!!!

# Material that you can use
# - the R help function
# - any printed material (open book)

# Notes
# - There are several ways how this exam can be solved.
# - What counts are ...
#   + that your program works
#   + that you follow the instructions
#   + that the program fulfills the requirements
# - We do not expect you to provide exactly the solution presented in class.

# Tip:
# - Use R to check your solution!


################################################
# 0: Your name    (2 points)                   #
################################################
# 0.1 Write your name below as a comment
# <Mokabbal Assiya>


################################################
# 1: R basics    (13 points)                   #
################################################
# 1.1 Create a vector v1 as defined in equation 1.1 on the Info Sheet. Do not use the c() command.
v1 = seq(10, 1, -1)
# 1.2 Create a vector v2 as defined in equation 1.2 on the Info Sheet. Do not use the c() command.
v2 = seq(3, 30, 3)
# 1.3 Create a matrix M as defined in equation 1.3 on the Info Sheet.
M = rbind(seq(1, 291, 10), seq(2, 292, 10), seq(3, 292, 10), seq(4, 294, 10), seq(10, 300, 10))
# 1.4 Print the first row of M
M[1,]
# 1.5 Display the last element of v1. Tell R to "display the last element of x",
#     regardless of the dimension of v1.
```

```r
tail(v1, 1)
# 1.6 Perform the calculations in equation 1.4 on the Info Sheet.
(12-(19-7))^1/5
log(1)+log(2)/((pi+1)/(pi-1))
log(sin(2)/exp(2))
# 1.7 Create a variable "u" with the value "ten to the power minus 5"
u = 10^(-5)
# 1.8 If you would now run the line
# print(U)
# you would get an error message. Which important concept of R is the reason
# for this error? Answer in 1 sentence as a comment.

"R is case sensitive"
################################################
# 2: Data and Logical conditions  (20 points)  #
################################################
# The file "Eudata.csv" contains data about the (still 28) EU countries.
# The columns are: County Name, Code, Capital, Accession (=Year of membership),
# Population, Area, GDP (in Million EUR), currency, IsEurozone (=1, if member)

# The following line loads the data into an R dataframe
# Hunt: Use Session/Set Working Directory/To Source File Location
Eudata <- read.table('Eudata.csv', sep=";",header=TRUE)

# 2.1 How many countries are there in the dataset?
nrow(Eudata)
# 2.2 Calculate the total population of the EU
sum(Eudata$Population)
# 2.3 Print the population of the smallest and largest EU country by Area
smallest= Eudata$Population[which.min(Eudata$Area)]
largest= Eudata$Population[which.max(Eudata$Area)]
# 2.4 Calculate the number of countries that are members of the Eurozone
sum(Eudata$Eurozone==TRUE)
# 2.5 Calculate the total GDP of all Eurozone members
sum(Eudata$GDP[Eudata$Eurozone == TRUE])
# 2.6 Calculate the GDP per capita in euros
#     (a) of the total EU, (b) of the Eurozone  (c) of the non-Eurozone EU

GDP_per_capita_EU = sum(Eudata$GDP/Eudata$Population)
GDP_Eurozone = sum(Eudata$GDP[Eudata$Eurozone]/Eudata$Population[Eudata$Eurozone])
GDP_non_EU = sum(Eudata$GDP[Eudata$Eurozone==FALSE]/Eudata$Population[Eudata$Eurozone==FALSE])

# 2.7 When was the EU founded?
#     Hint: this must be the earliest year in which any country became a member
```

```r
min(Eudata$Accession)
# 2.8 Calculate the number of EU founding members
sum(Eudata$Accession == 1953)
# 2.9 Only now you discover that the data set still contains the UK.
#     Permanently remove the UK from the dataframe "Eudata"
Eudata = Eudata[Eudata$CountyName !="United Knigdom", ]
# 2.10 You also discover that Bulgaria actually joined the Euro on 1st January 2026.
#      Permanently update the dataframe "Eudata" accordingly.
Eudata = Eudata[Eudata$Eurozone == "Bulgaria" ]


###############################################
# 3: Simulation and probability   (15 points)  #
###############################################
# 3.1 Use R to produce one roll of a dice.
dice = sample(1:6, 1, replace= TRUE)
# 3.2 Create a vector called "k" that contains 1000 rolls of a dice
k= sample(1:6, 1000, replace = TRUE)
# 3.3 Using "k" from (3.2), estimate the probability of obtaining
#     a "4" or "5" from a dice
mean(k==4 | k==5)
# 3.4 Create a vector called "m" that contains 1000 (different) rolls of a dice
#     Using "k" and "m" from (3.2) and (3.4), estimate the expected value
#     and variance of the random variable z = 2k-m
m = sample(1:6, 1000, replace = TRUE)
z = 2*k-m
mean(z)
# 3.5 Assume the yearly stock return to be normally distributed with a mean of 0.12 and
#     a standard deviation of 0.2. Create a variable "stock" with 100 draws of stock returns
returns_stock = rnorm(100, 0.12, 0.2)
# 3.6 What is the probability of a negative stock return?
#     Answer this question ...
#     (a) by using the variable "stock" from (3.5)
#     (b) by calculating the (theoretical) probability for a normal distribution
mean(returns_stock > 0)
pnorm(0, 0.12, 0.2)
###############################################
# 4: Functions and Optimization     (25 points)#
###############################################
# 4.1 Create the function f(x)=x^2 in R
f = function(x){x^2}
# 4.2 Calculate the value of f for x=1
f(1)
# 4.3 Create a plot of the function for the interval [-2, 2]
#     If in doubt, type "?plot" to get the help file for the function
curve(f, -2, 2)
# 4.4 Numerically, by using R, find the location of the minimum using the
```

```
optim
#     function. Store the result of your minimization (the location of the minimum)
#     in a variable called xmin
neg_f = function(x){
  -f(x)
}
res <- optim(par=0, fn= neg_f, method = "Brent", lower = -2, upper = 2)
xmin <- res$par
# 4.5 Now try to find the location of the minimum by implementing a grid search
#     yourself. Choose N=100 grid points. Search in an interval between -2 and 2.
#     Store the result of your minimization (the location of the minimum) in a
#     variable called xmin_grid

# 4.6 Answer in a short comment (<= 2 sentences). The results from (4.4) and (4.5)
#     are not identical. Why?


y###################################################
# 5: Functions and algorithms    (25 points)       #
####################################################
# 5.1  The Luhn Algorithm is used to check whether a credit card number is valid. It goes
#      like this
#      a) process individual digits from right to left
#      b) leave digits number 1,3,5 etc (counted from right) unchanged
#      c) multiply digits 2,3,6 etc  (counted from right) by 2
#      d) if a digit (after multiplying by 2) is larger than 9, subtract 9
#      e) calculate the sum of all (processed) digits
#      IF the result can be divided by 10, the number is a valid credit card number
#
#      Example: 63487
#      Right to left: do not change 7, multiply 8 by 2 and subtract 9, do not change 4,
#                     multiply 3 by 2 and do not subtract anything, do not change 6
#                     7 + 8*2-9 + 4 + 3*2 + 6 = 30 --> the number is valid
#      Hint: The operation x %% y yields the remainder of the division x/y.
#            For instance, 7 %% 4 gives 3

# Write a function called checkLuhn that takes as argument a vector of individual digits
# and returns TRUE if the number is a valid number and FALSE if it is not valid.
```

```r

# The follwing line takes a *valid* credit card number and creates a vector with single digits.
# You can use this to test your function
# Hint: digits contains the numbers from LEFT to RIGHT, while the Luhn algorithm works from
# right to left.
digits <- as.numeric(unlist(strsplit("5019717010103742","")))



```