```
#############
#
# Task 1
#
#############

a <- 1:3 # or a <- c(1,2,3); a <- seq(1,3,1); a <- c(1L,2L,3L)
# Note, that a <- 1:3 and a <- c(1L,2L,3L) will generate integer vectors.
a

(b <- c(3,-5,0))

A <- matrix(c(3,1.5,1/8,5,-pi,-6,-1,exp(2.5),9), nrow = 3)
# A <- matrix(c(3,1.5,1/8,5,-pi,-6,-1,exp(2.5),9), ncol = 3)
# A <- matrix(c(3,1.5,1/8,5,-pi,-6,-1,exp(2.5),9), nrow = 3, ncol = 3)
# A <- matrix(c(3,5,-1,1.5,-pi,exp(2.5),1/8,-6,9), byrow = TRUE, nrow = 3)
# Note: R is column-major, hence byrow = FALSE by default
print(A)

(B <- c(-1,2,-3) %*% t(c(-1,2,-3)))
# B <- matrix(c(1,-2,3,-2,4,-6,3,-6,9), nrow = 3)


# (ii)

(A %*% a)
(B %*% b)
(A %*% A)
(A %*% B)
(t(a) %*% b) # or a%*%b


# (iii)

# Simply use +,-,*,/ as component-wise operators
(b + a); (b - a); (b * a); (b / a)

(B + A); (B - A); (B * A); (B / A)


# (iv)

(B.prime.1 <- rbind(B[1,] + B[2,], B[3,]))
(B.prime   <- cbind(B.prime.1[,1] + B.prime.1[,2], B.prime.1[,3]))
# B.prime <- matrix(c(sum(B[1:2,1:2]),sum(B[1:2,3]), sum(B[3,1:2]), B[3,3]), nrow = 2)


# (b)

(x <- c(5, 2, 6, 4, 1, 2, 2, 5, 4, 4, 6, 4, 2, 5, 5, 3, 6, 1, 4, 5))

# (y <- cut(x, breaks = c(0,2,4,6)))
(y <- cut(x, breaks = c(0,2,4,6), labels = 1:3))
# convertion to numeric
(y <- as.numeric(y))

# In our special case, other solutions are possible, e.g.
# (y <- ceiling(x/2))


#############
#
# Task 2
#
#############

# (a)

v1 <- c(TRUE, TRUE, FALSE, TRUE, FALSE); v2 <- 1:6; v3 <- 5:10
v1; v2; v3


# (b)

sum(v1)
prod(v1)
# TRUE is interpreted as 1 and FALSE is interpreted as 0

(v4 <- as.numeric(v1))
# (v4 <- v1 + 0)


# (c)
# (i)

res <- 0
for(i in seq_along(v2)) {
  res <- res + (v2[i] * v3[i])^i
}
res
# also possible as argument for "for"-loop:
# i in 1:length(v2)
# i in seq_len(length(v2))
# One could also use v3 instead of v2 in the argument.

# Note the difference in case of length(v3) = 0.


# (ii)

(res <- sum((v2 * v3)^(1:6)))


# (d)

which(v2[-6] > v4)[1]
# sum(cumsum((v2[-6] > v4)) == 0) + 1



min.len <- min(length(v2), length(v4))
i <- 1

while((i < min.len) && (v2[i] <= v4[i])){
  i <- i+1
}
print(i)


# (e)

example.function <- function(vec1, vec2) {
  if(length(vec1) == length(vec2)) {
    res <- sum((vec1 * vec2)^(1:length(vec1)))
  } else {
    min.len <- min(length(vec1), length(vec2))
    res <- which(vec1[1:min.len] > vec2[1:min.len])[1]
    if(is.na(res)) res <- Inf
  }
  return(res)
}

#############
#
# Task 3
#
#############

# (a)
# (i)

credits.data = read.table("credits.wsv", header=TRUE)

# (ii)

print(credits.data$amount)

str(credits.data$amount)

credits.data[,2] # second column
credits.data[2,] # second row

credits.data[,-(1:8)] # everything without first eight columns

credits.data[-(1:8),] # everything without first eight rows

credits.data[1:6,] # first six rows
# head(credits.data)

credits.data[45:50,] # last six rows
# tail(credits.data)


# (iii)

mean(credits.data$amount)
median(credits.data$amount)

summary(credits.data$amount)


# (b)
# (i)

my.sd <- function(data, corrected = TRUE) {
  TSS <- sum((data - mean(data))^2)
  res <- sqrt(TSS/(length(data) - corrected))
  return(res)
}

# (ii)

my.sd(credits.data$amount)
my.sd(credits.data$amount, corrected = FALSE)

sd(credits.data$amount) # uses the corrected variant by default



#############
#
# Task 4
#
#############


# sample size n=10, 50, 100
for(n in c(10, 50, 100)){
  print("Sample size:")
  print(n)
  # (a)

  # generate random numbers, remember: sd=sqrt(sigma^2)
  X = rnorm(n, mean=5, sd = 2)

  # (b)

  # proportion of random numbers in the interval
  h1 = mean(3 <= X & X <= 7)
  h2 = mean(5 <= X & X <= 9)
  # TRUE is interpreted as 1 and FALSE is interpreted as 0

  # probability of the normal distribution
  p = pnorm(c(7,9), mean = 5, sd = 2) - pnorm(c(3,5), mean = 5, sd = 2)

  print(list(proportion=c(h1, h2), probability = p))

  # (c)

  # empirical quantiles
  q1 = quantile(X, probs=c(0.3, 0.5, 0.75))
  # quantiles of the normal distribution
  q2 = qnorm(c(0.3, 0.5, 0.75), mean=5, sd = 2)

  print(list(emp_quantile=q1, quantile=q2))
}
```