1. Solve the maximum expected returns optimization exercise in section 3 of Lecture 11 (that is, reproduce the solution presented in the lecture). The last line of your R program should display the portfolio weights vector w.

```
G <- function(x, mu, Sigma, sigmaP2)
{
n <- length(mu)
c(mu + rep(x[n+1], n) + 2*x[n+2]*(Sigma %*% x[1:n]),

sum(x[1:n]) - 1,
t(x[1:n]) %*% Sigma %*% x[1:n] - sigmaP2)
}

DG <- function(x, mu, Sigma, sigmaP2)
{
n <- length(mu)
grad <- matrix(0.0, n+2, n + 2)
grad[1:n, 1:n] <- 2*x[n+2]*Sigma

grad[1:n, n+1] <- 1
grad[1:n, n+2] <- 2*(Sigma %*% x[1:n])
grad[n+1, 1:n] <- 1
grad[n+2, 1:n] <- 2*t(x[1:n]) %*% Sigma
grad
}

x <- c(rep(0.5, 5), 1, 1)
u <- rep(1, length(x))
mu <- c(0.08, 0.10, 0.13, 0.15, 0.20)
Sigma <- matrix(c(0.019600,-0.007560, 0.012880,0.008750,-0.0098,-0.007560,
```

```
0.032400, -0.004140, -0.009000, 0.009450, 0.012880, -0.004140,

0.052900, 0.020125, 0.020125, 0.008750,-0.009000, 0.020125,

0.062500, -0.013125, -0.009800, 0.009450, 0.020125, -0.013125,0.122500)

,nrow = 5,ncol = 5, byrow = TRUE)



while(sqrt(sum(u^2)) / sqrt(sum(x^2)) > 1e-6) {

        u <- solve(DG(x, mu, Sigma, 0.25^2),

        G(x, mu, Sigma, 0.25^2))

        x <- x - u

}


DG(x, mu, Sigma, sigmaP2)[1:5, 1:5]

eigen(DG(x, mu, Sigma, sigmaP2)[1:5, 1:5])$values

t(x[1:5]) %*% mu

x[1:5]
```

2. Using the same expected returns vector μ and covariance matrix Σ from the previous exercise, solve the optimization problem

minimize: wTΣw
subject to: eTw = 1

μTw = μ2

where μ2 is the expected return computed in the previous exercise. Again, the last line of your R program should display the portfolio weights vector w. Compare with the previous answer.

```
G <- function(x, mu, Sigma)

{

n <- length(mu)

c(2*(Sigma %*% x[1:n])+ rep(x[n+1], n) + x[n+2]*mu,

sum(x[1:n]) - 1,
```

```
t(mu)%*%x[1:n] - 0.1991514)
}

#Sigma e mu
#e^t 0 0
#mu^T
DG <- function(x, mu, Sigma)
{
n <- length(mu)
grad <- matrix(0.0, n+2, n + 2)
grad[1:n, 1:n] <- 2*Sigma
grad[1:n, n+1] <- 1
grad[1:n, n+2] <- mu
grad[n+1, 1:n] <- 1
grad[n+2, 1:n] <- t(mu)
grad
}

x <- c(rep(0.5, 5), 1, 1)
u <- rep(1, length(x))
mu <- c(0.08, 0.10, 0.13, 0.15, 0.20)
Sigma <- matrix(c(0.019600,-0.007560, 0.012880,0.008750,-0.0098,-0.007560,
0.032400, -0.004140, -0.009000, 0.009450, 0.012880, -0.004140,
0.052900, 0.020125, 0.020125, 0.008750,-0.009000, 0.020125,
0.062500, -0.013125, -0.009800, 0.009450, 0.020125, -0.013125,0.122500)
,nrow = 5,ncol = 5, byrow = TRUE)


while(sqrt(sum(u^2)) / sqrt(sum(x^2)) > 1e-6) {
```

```
	u <- solve(DG(x, mu, Sigma),
	G(x, mu, Sigma))
	x <- x - u
}
#Check second order condition - it is positive definite matrix
#therefore it is a minimum var
eigen(DG(x, mu, Sigma)[1:5, 1:5])$values
#portfolio weights
x[1:5]
```

3. Write a function to compute the covariance matrix for a given collection of sample data

```
covar <- function(matrix)
{
	len = nrow(matrix)
	e = c(rep(1,times = len))
	A = diag(len) - e*e/len
	x <- colMeans(matrix, na.rm = FALSE, dims = 1)
	#x <- matrix(c(x,x,x),nrow = 3, ncol = 3, byrow = TRUE)
	#Using (x-x_tilde)^T(x-x_tilde)
	#1/(len-1)*(t(matrix-x)%*%(matrix-x))
	#x_tilde <- matrix - x*e
	#1/(len-1)*(t(x_tilde)%*%x_tilde)
	Ax <- (diag(len) - e*e/len)%*%matrix
	1/(len-1)*(t(Ax)%*%Ax)
}
```