Problem 1

For part(c), I used the following code to compare different methods.

```
X <- cbind(1, as.matrix(longley[, -7]))
y <- as.vector(longley[, 7])
X.b <- as.bigq(round(1000 * X))/as.bigq(1000)
y.b <- as.bigq(round(1000 * y))/as.bigq(1000)
as.vector(lm.fit.exact(X.b, y.b) - lm.fit(X, y)$coefficients)
lm.fit.exact(X.b, y.b) - lm.fit.chol(X, y)
lm.fit.exact(X.b, y.b) - lm.fit.chol(X[, -1], y, center = TRUE)
```

The results are as follows.

```
> as.vector(lm.fit.exact(X.b, y.b) - lm.fit(X,y)$coefficients)
[1]   -5.911716e-12     6.002143e-16   -9.714451e-17   -1.734723e-17   -3.469447e-17
-1.422473e-15  3.108624e-15
> lm.fit.exact(X.b, y.b) - lm.fit.chol(X,y)
[1]   -3.492773e-05     2.954203e-10   -9.547554e-10   -1.460293e-10   -4.604652e-11
2.386142e-09  1.793108e-08
> lm.fit.exact(X.b, y.b) - lm.fit.chol(X[, -1],y,center=TRUE)
[1]   -2.733032e-10     1.076049e-14   -9.915679e-15   -1.464107e-15   -3.989864e-16
4.453382e-14  1.389999e-13
```

According to the errors, it is very clear that QR is better than Cholesky. Though the mean-centered data dose improve the accuracy of Cholesky, QR is still superior to Cholesky.

Problem 2

For part(c), I used the following code to test my functions.

```
#n=9
y <- c(rep(0, 3), rep(1, 3), rep(-1, 3))
system.time(for (i in 1:1000) loglikD(y, 0.5))
system.time(for (i in 1:1000) loglikS(y, 0.5))

#n=30
y <- c(rep(0, 10), rep(1, 10), rep(-1, 10))
system.time(for (i in 1:1000) loglikD(y, 0.5))
system.time(for (i in 1:1000) loglikS(y, 0.5))

#n=180
y <- c(rep(0, 60), rep(1, 60), rep(-1, 60))
system.time(for (i in 1:1000) loglikD(y, 0.5))
system.time(for (i in 1:1000) loglikS(y, 0.5))

#n=300
y <- c(rep(0, 100), rep(1, 100), rep(-1, 100))
system.time(for (i in 1:1000) loglikD(y, 0.5))
system.time(for (i in 1:1000) loglikS(y, 0.5))
```

The results are as follows.

n	loglikD	loglikS
9	0.094	1.533
30	0.106	1.536
180	1.460	1.607
300	6.282	1.653

As we can see, dense method is very sensitive to the matrix dimension. If the matrix is small, it performs better than sparse method. However, if the matrix is large, it becomes very slow. In contrast, the sparse method is not sensitive to the matrix dimension. It is a little bit slow for small matrices but very efficient for large ones.