Exercise 2

The LOOCV estimate can be automatically computed for any generalized linear model using the `glm()` together with the `cv.glm()` function. We've used the `glm()` before for logistic regression by passing in the `family="binomial"` argument. But if we use `glm()` to fit a model without passing in the family argument, then it performs linear regression, just like the `lm()` function.

```
##############    Leave-One-Out Cross-Validation    ##########################


glm.fit=glm(mpg~horsepower,data=Auto)
coef(glm.fit)

lm.fit=lm(mpg~horsepower,data=Auto)
coef(lm.fit)

library(boot)

# Without specifying "family" as a parameter, glm() falls back to linear regression.
# We use it here, because it comes with a function for cross validation
glm.fit=glm(mpg~horsepower,data=Auto)

cv.err=cv.glm(Auto,glm.fit) # computing the LOOCV prediction Error. If dont set the parameter K, cv.glm defaults to LOOCV
cv.err$delta[1] # The estimate for the test error is stored in delta[1].
```

Here we can see that both functions indeed do the same.

Exercise 2

The LOOCV estimate can be automatically computed for any generalized linear model using the `glm()` together with the `cv.glm()` function. We've used the `glm()` before for logistic regression by passing in the `family="binomial"` argument. But if we use `glm()` to fit a model without passing in the family argument, then it performs linear regression, just like the `lm()` function.

```
# Now we calculate the LOOC test error for polynomial regression models of degree 1,...,10.
# It takes a bit to evaluate...

cv.error=rep(0,10) # initialising the LOOCV error vector
for (i in 1:10){
  glm.fit=glm(mpg~poly(horsepower,i),data=Auto)
  cv.error[i]=cv.glm(Auto,glm.fit)$delta[1]
}
cv.error

# Plot the results
x <- seq(1,10,1)
plot(cv.error~x, col="blue3")
lines(cv.error~x, col="blue3")
```