```
# Mini Project 1
# Matthew Lynn

# First we want to bring in our data sets for training and testing----

train = read.csv("1-training_data.csv", header = T)
test = read.csv("1-test_data.csv", header = T)

# Lets take a look at the data
# First we use the head() to check a snippet of our columns and what kind of obs we are dealing with
# Then we use str() to determine the data types of each column
# Based on head() and str() we see that x.1 and x.2 are quantitative and y is qualitative
# Summary() gives us some quick stats and shows us that our y variable is 50/50 with yes and no obs

head(train)
str(train)
summary(train)

# Now we want to look at a scatterplot of every variable
# Also we'd like to see what our correlation matrix looks like
# It appears that x.1 and x.2 have a negative trending correlation

pairs(train)
round(cor(train[,-3]), 3)

# Lets use y as the response variable and the remaining variables as predictors

attach(train)

# want to double check that the dimensions and columns are correct

train.x = cbind(x.1, x.2)
dim(train.x)
head(train.x)

train.y = y
head(train.y)
table(train.y)

# Now lets plot out or training data

plot(train.x, xlab = "x.1", ylab = "x.2", col = ifelse(train.y == "yes", "green", "red"))

# Lets set up our test data

attach(test)

test.x = cbind(x.1, x.2)
test.y = y

# A quick glance at the Global Environment window lets use know that all our dimensions are good to go
# Now we can answer some questions

# Question 1 part a----
# The goal for question 1 is to fit KNN for K = 1:30 by 1 and 35:100 by 5

library(class)

# Lets set up a variable so that we can fit KNN for several values of K
# Then we set up our error rates to be fed into a loop along ks
# The names part forces the column headers of the error rates and ks to be the same
# This way they are easier to read in a table as well as functions that call them against each other

ks = c(seq(1, 30, by = 1), seq(35, 100, by = 5))
nks = length(ks)
err.rate.train = numeric(length = nks)
err.rate.test = numeric(length = nks)
names(err.rate.train) = names(err.rate.test) = ks

# Now we iterate along each value of ks to produce a set of error rates

for (i in seq(along = ks)) {
  set.seed(160230)
  mod.train = knn(train.x, train.x, train.y, k = ks[i])
  set.seed(160230)
  mod.test = knn(train.x, test.x, train.y, k = ks[i])
  err.rate.train[i] = 1 - sum(mod.train == train.y)/length(train.y)
  err.rate.test[i] = 1 - sum(mod.test == test.y)/length(test.y)
}

# here we plot the test and training errors together

plot(ks, err.rate.train, xlab = "Number of nearest neighbors", ylab = "Error rate",
     type = "b", ylim = range(c(err.rate.train, err.rate.test)), col = "blue", pch = 20)
lines(ks, err.rate.test, type="b", col="purple", pch = 20)
legend("bottomright", lty = 1, col = c("blue", "purple"), legend = c("training", "test"))

# Now we want to find the optimal k using a min function

result <- data.frame(ks, err.rate.train, err.rate.test)
result[err.rate.test == min(result$err.rate.test), ]

# Now we want to make a decision boundary

n.grid <- 50
x1.grid <- seq(f = min(train.x[, 1]), t = max(train.x[, 1]), l = n.grid)
x2.grid <- seq(f = min(train.x[, 2]), t = max(train.x[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)

# we set the optimal k found above and modify the attribute "prob" from mod.opt to feed us the "yes" obs only
# to be used on the contour that lays over x.1 and x.2

k.opt <- 80
set.seed(1)
mod.opt <- knn(train.x, grid, train.y, k = k.opt, prob = T)
prob <- attr(mod.opt, "prob") # prob is voting fraction for winning class
```

```
prob <- ifelse(mod.opt == "yes", prob, 1 - prob) # now it is voting fraction for Direction == "Up"
prob <- matrix(prob, n.grid, n.grid)

plot(train.x, col = ifelse(train.y == "yes", "green", "red"))
contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
```