Q2.

```
data(crabs)
head(crabs)
```

```
##   sp sex index   FL  RW   CL   CW  BD
## 1  B   M     1  8.1 6.7 16.1 19.0 7.0
## 2  B   M     2  8.8 7.7 18.1 20.8 7.4
## 3  B   M     3  9.2 7.8 19.0 22.4 7.7
## 4  B   M     4  9.6 7.9 20.1 23.1 8.2
## 5  B   M     5  9.8 8.0 20.3 23.0 8.2
## 6  B   M     6 10.8 9.0 23.0 26.5 9.8
```

```
set.seed(6789)
inTrain <- createDataPartition(crabs$sp, p = 0.8, list = FALSE)
training <- crabs[inTrain,]
testing <- crabs[-inTrain,]
```

(a)

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.3
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

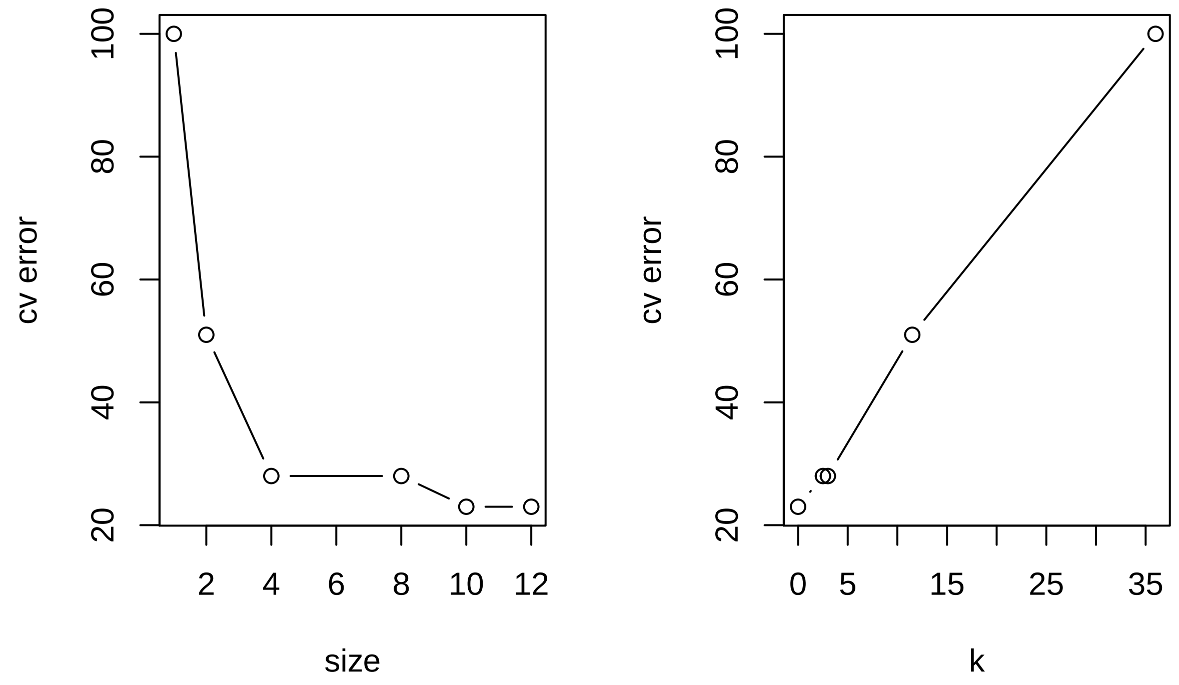
```
set.seed(10)
tree.crabs=tree(sp~. -index, training)
cv.crabs=cv.tree(tree.crabs,FUN=prune.misclass)
names(cv.crabs)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.crabs
```

```
## $size
## [1] 12 10  8  4  2  1
##
## $dev
## [1]  23  23  28  28  51 100
##
## $k
## [1] -Inf  0.0  2.5  3.0 11.5 36.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

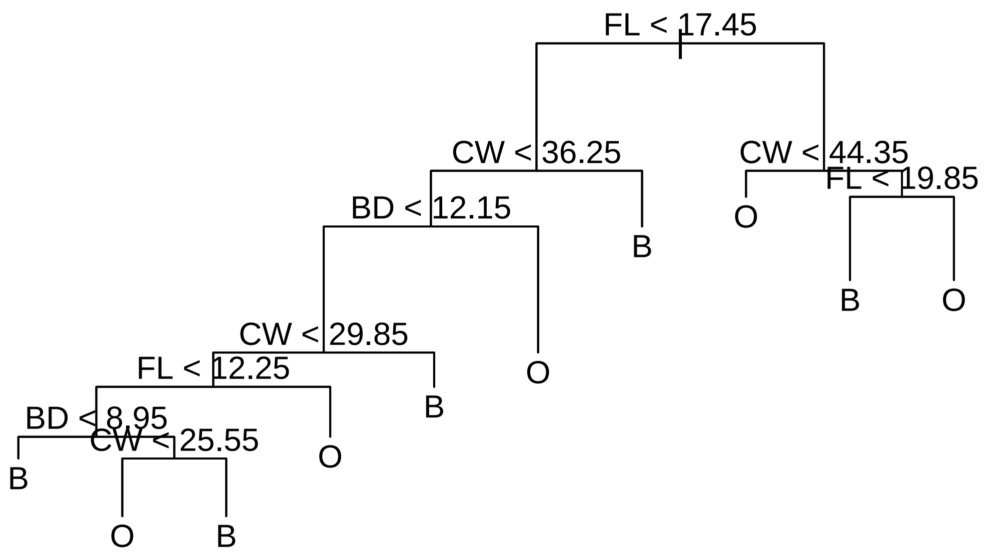
```
par(mfrow=c(1,2))
plot(cv.crabs$size,cv.crabs$dev / length(train),ylab="cv error", xlab="size",type="b")
plot(cv.crabs$k, cv.crabs$dev / length(train),ylab="cv error", xlab="k",type="b")
```



Hence, the lowest test error is obtained when size = 10

#plot

```
prune.crabs=prune.misclass(tree.crabs,best=10)
plot(prune.crabs)
text(prune.crabs,pretty=0)
```



```
#test error
tree.pred=predict(prune.crabs,testing,type="class")
table(tree.pred,testing$sp)
```

```
##
## tree.pred  B  O
##         B 18  3
##         O  2 17
```

```
(test.err<-(3+2)/40) #test error
```

```
## [1] 0.125
```

```
#train error
tree.pred_train=predict(prune.crabs,training,type="class")
table(tree.pred_train,training$sp)
```

```
##
## tree.pred_train  B  O
##               B 78  2
##               O  2 78
```

```
(train.err<-(2+2)/160) #train error
```

```
## [1] 0.025
```

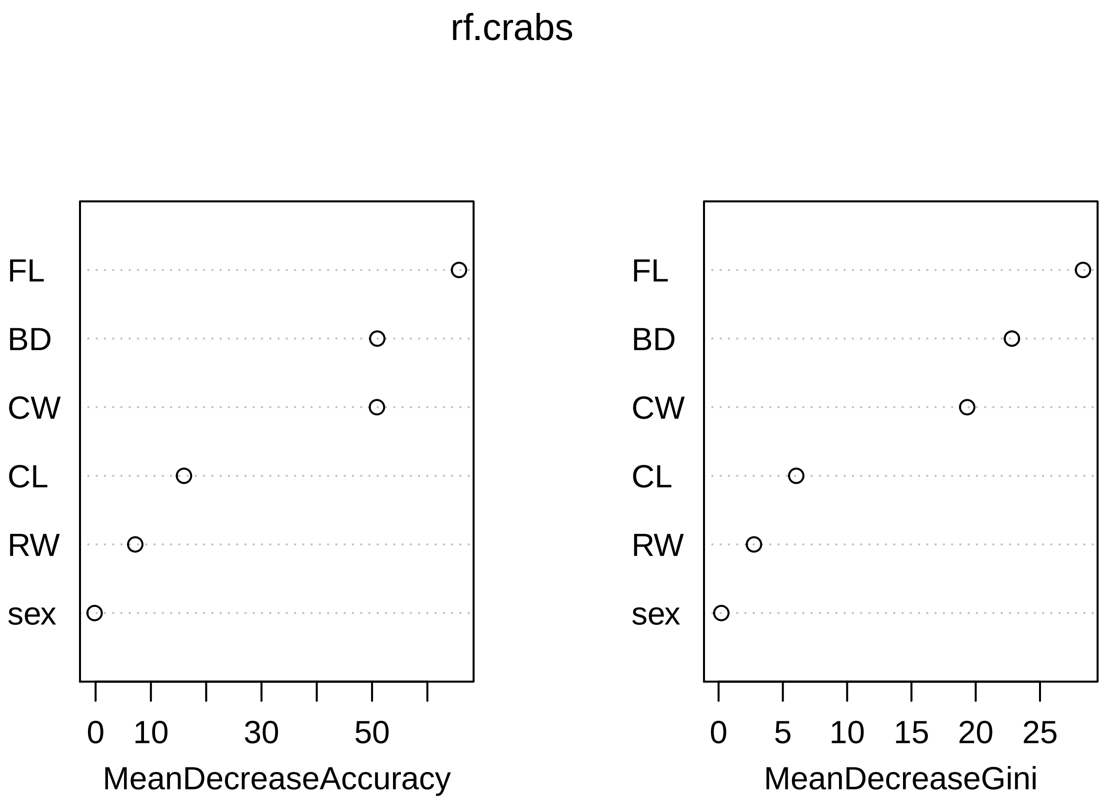
(b)

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf.crabs=randomForest(sp~.-index,data=crabs,subset=inTrain,mtry = 5,importance=TRUE)
```

```
#importance plot
varImpPlot(rf.crabs)
```



The two most important variables are FL and BD, however, the result of single tree shows that FL and CW are the two most important variables, which indicates a difference.

```
#train error
rf.train_pred = predict(rf.crabs, training)
table(rf.train_pred,training$sp)
```

```
##
## rf.train_pred  B  O
##              B 80  0
##              O  0 80
```

```
(train.err<-(0+0)/160) #train error
```

```
## [1] 0
```

```
#test error
rf.test_pred = predict(rf.crabs, testing)
table(rf.test_pred,testing$sp)
```

```
##
## rf.test_pred  B  O
##            B 15  2
##            O  5 18
```

```
(test.err<-(5+2)/40) #test error
```

```
## [1] 0.175
```

(C)

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.3

## Loaded gbm 2.1.8
```

```
library(adabag)
```

```
## Warning: package 'adabag' was built under R version 4.0.3

## Loading required package: rpart

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: doParallel

## Warning: package 'doParallel' was built under R version 4.0.3

## Loading required package: iterators

## Loading required package: parallel
```

```
crabs$sp01 = ifelse(crabs$sp=="B", 1, 0)
set.seed(6789)
inTrain <- createDataPartition(crabs$sp01, p = 0.8, list = FALSE)
training <- crabs[inTrain,]
testing <- crabs[-inTrain,]

#crabs.boostcv<-boosting.cv(sp~.-index-sp01,data = training, v = 5, mfinal = 1000)
#The cross-validation step is too slow, so this step is skipped in the result

adaboost.fit = gbm(sp01~.-index-sp, data = training , distribution="adaboost", n.trees=1000)
```

```r
#train error
probs.adaboost = predict(adaboost.fit, training, n.trees = 1000, type = 'response')
pred.adaboost = ifelse(probs.adaboost > 0.5, 1, 0)
train_err = mean(pred.adaboost!=training$sp01)
print(train_err)
```

```
## [1] 0.0125
```

```r
#test error
probs.adaboost = predict(adaboost.fit, testing, n.trees = 1000, type = 'response')
pred.adaboost = ifelse(probs.adaboost > 0.5, 1, 0)
test_err = mean(pred.adaboost!=testing$sp01)
print(test_err)
```

```
## [1] 0.075
```

(d) The results of Adaboost reports the second lowest train error and lowest test error. Hence, for this dataset, we should choose adaboost as our method. The results are non-consistent across methods.