Exercise 3 - Pruning the regression tree (Hitters data)

```
#####  Cost-Complexity Pruning
#  Goal: prune the trees to avoid high variance and overfitting.
#  Positive effects:
#     - smaller *test* errors (due to less overfitting).
#     - higher interpretability (due to smaller trees).


# use cross-validation to find the optimal parameter \alpha for cost-complexity pruning
set.seed (3)
?cv.tree
cv.Hitters =cv.tree(tree.salaryHitters)
    # Runs a K-fold cross-validation experiment to find the number of
    # misclassifications as a function of the cost-complexity parameter \alpha.

cv.Hitters
    # $k: cost-complexity parameter (corresponds to \alpha)
          # Notice that \alpha is increasing (corresponding to the pruning sequence).
    # $size: number of terminal nodes of each tree
          # Notice that the size is decreasing (corresponding to the pruning sequence).
    # $dev: *cross-validation* error rate
          # The full tree (with size 9, i.e. 9 terminal nodes) has lowest cross-validation error.

# plot the cross-validation error-rate as a function of both size and \alpha (k):
par(mfrow=c(1,2))
plot(cv.Hitters$size, cv.Hitters$dev, type="b") # type="b": plot both, points and lines
plot(cv.Hitters$k, cv.Hitters$dev, type="b")
par(mfrow=c(1,1))

# we do not need to prune the tree, since the full tree has minimal cross-validation error
# yet, to show how to prune, here's the code:
prune.salaryHitters <- prune.tree(tree.salaryHitters, best=6)
    # prune.tree determines the nested cost-complexity sequence
    # best=6: get the 6-node tree in the cost-complexity sequence

# Plot the pruned regression tree
plot(prune.salaryHitters)
text(prune.salaryHitters, cex=0.75) # cex: set character size to 0.75
```