1 Functions/Options Learned

`Timing`	`AbsoluteTiming`	`DownValues`	`RecursionLimit`

2 Problems

From electronic text

1. Problem 11.1 (This problem uses a technique called memoization, which is not explained in the text in any way. This concept is explored further in the lesson associated with the homework.)
2. Exercise 11.1 (Mimic what the book did and use memoization on this problem. Generate at least 15 distinct primes using this recurrence.)
3. Problem 11.2 (This problem also uses memoization. Additionally, there is a typo, it should read "Check that t(100)=10000.")
4. Problem 11.3
5. Problem 11.4
6. Exercise 11.2
7. Problem 11.5 (This problem also uses memoization.)

From lesson on memoization

8. Exercise 1
9. Exercise 2
10. Exercise 3
11. Exercise 4
12. Quicksort (see `http://en.wikipedia.org/wiki/Quicksort`) is a divide and conquer algorithm for sorting a list of numbers. Quicksort first divides a large list into two smaller sublists: the "small" elements and the "big" elements and then recursively sorts the sublists. Here are the steps involved in the quicksort algorithm:
    - Pick an element, called a pivot, randomly from the list.
    - Of the remaining non-pivot elements, create two new sublists, a list of numbers smaller than the pivot element and a list of numbers greater than the pivot element.
    - Recursively apply the above steps to the sublists described above separately and return the list obtained by joining the outputs of the the recursive calls with the pivot element in the correct order.

    (a) Create a recursive function called `Quicksort[L_List]` using `Module` that takes a list `L` as an input and returns the list in ascending order. You may not use any built-in sorting operations.

    (b) Demonstrate your function on a list of random integers between 1 and 10000 of length 100 to prove that it works correctly.

13. The partial Schensted insertion algorithm takes a permutation of $\{1, 2, \ldots, n\}$ and returns a list of lists satisfying the property that if you were to stack the lists on top of each other, they would be increasing in both rows and columns. It is accomplished by starting with the empty list and then successively inserting each element of the permutation (in order) into the list using row-insertion. The following describes how row-insertion works for inserting an integer x into the list L.

   - If L is empty, return $\{\{x\}\}$.
   - If x is larger than every element in the first list of L, then return the list obtained by appending x to the first list of L.
   - If there exists a y in the first list of L such that $y > x$, choose the smallest such y and return the list whose first element is the same as the first of L except with x in place of y, and whose remaining lists are the output of inserting y into what remains after removing the first list from L (which could be empty).

   To ensure that you understand how this works, compute the partial Schensted insertion algorithm by hand on the permutation $\{2, 1, 3, 6, 5, 7, 10, 8, 9, 4\}$. The answer should be $\{\{1, 3, 4, 7, 8, 9\}, \{2, 5, 10\}, \{6\}\}$, which you could view using `TableForm` (shown below) to see how it is increasing in rows (from left to right) and in columns (from top to bottom).

```
1  3   4   7  8  9
2  5  10
6
```

   (a) Implement the partial Schensted insertion algorithm by writing a function called `Schensted[P_List]` that takes a permutation `P` of $\{1, 2 \ldots, n\}$ as an input. This function should make use of the built-in function `Fold` as well as a recursive function (written by you) called `RowInsert[L_List,x_Integer]` that performs the row insertion of `x` into `L` described above.

   (b) Demonstrate your function on the permutation $\{2, 8, 7, 12, 6, 1, 11, 4, 5, 3, 9, 10\}$ to prove that it works correctly.

   (c) For each $1 \leq i \leq 6$, how many permutations of $\{1, 2, \ldots, 6\}$ result in a list of length i under this algorithm? Check out the function `Tally` or `Counts`.

14. Extend your code from Exercise **??** to perform the full Schensted algorithm, see `http://en.wikipedia.org/wiki/Robinson-Schensted_correspondence`. The full Schensted algorithm will take a permutation of $\{1, 2, \ldots, n\}$ and return a pair of lists of lists. The first list in the pair is precisely the output of the partial Schensted algorithm described in Exercise **??**. The second list of lists will have the same "shape" as the first and be filled with the integers $\{1, 2, \ldots, n\}$ corresponding to the order in which the "cells" are created as you successively insert each element of the permutation using row-insertion. For example, we saw that the permutation $\{2, 1, 3, 6, 5, 7, 10, 8, 9, 4\}$ mapped to to $\{\{1, 3, 4, 7, 8, 9\}, \{2, 5, 10\}, \{6\}\}$ under the partial Schensted algorithm. Under the full Schensted algorithm, the image would be

$$\{\{\{1, 3, 4, 7, 8, 9\}, \{2, 5, 10\}, \{6\}\}, \{\{1, 3, 4, 6, 7, 9\}, \{2, 5, 8\}, \{10\}\}\}\,.$$

Here is what they look like using `TableForm`.

$$\left\{ \begin{array}{llllll} 1 & 3 & 4 & 7 & 8 & 9 \\ 2 & 5 & 10 & & & \\ 6 & & & & & \end{array} , \begin{array}{llllll} 1 & 3 & 4 & 6 & 7 & 9 \\ 2 & 5 & 8 & & & \\ 10 & & & & & \end{array} \right\}$$

Note that both lists have the same shape and are increasing in both rows and columns (this will always be the case). Implement the full Schensted insertion algorithm by writing a function `FullSchensted[L_List]`. Test your function by implementing the full Schensted algorithm on the permutation $\{5, 12, 6, 7, 4, 2, 8, 11, 10, 13, 15, 3, 14, 9, 1\}$.

15. **Extra Credit:** It turns out that the full Schensted algorithm is a bijection between permutations of $\{1, 2, \ldots, n\}$ and pairs of lists consisting of the integers $\{1, 2, \ldots, n\}$ of the same shape which are increasing in both rows and columns. Thus, the Schensted algorithm is invertible. Implement the inverse mapping by writing a function `InverseSchensted[P_List,Q_List]` that will take a pair of lists $\{$`P,Q`$\}$ and return the permutation whose image under the full Schensted algorithm is $\{$`P,Q`$\}$. Test your function by computing the inverse Schensted algorithm of the pair

$$\{\{\{1, 2, 4, 8, 11\}, \{3, 6, 7, 10\}, \{5, 9\}\}, \{\{1, 4, 6, 9, 11\}, \{2, 5, 7, 10\}, \{3, 8\}\}\}$$

or in `TableForm`,

$$\left\{ \begin{array}{lllll} 1 & 2 & 4 & 8 & 11 \\ 3 & 6 & 7 & 10 & \\ 5 & 9 & & & \end{array} , \begin{array}{lllll} 1 & 4 & 6 & 9 & 11 \\ 2 & 5 & 7 & 10 & \\ 3 & 8 & & & \end{array} \right\}.$$