**58093 String Processing Algorithms (Autumn 2014)**
Exercises 3 (November 11)

1. Prove

   (a) Lemma 1.14: For $i \in [2..n]$, $LCP_{\mathcal{R}}[i] = lcp(S_i, \{S_1, \ldots, S_{i-1}\})$.

   (b) Lemma 1.15: $\Sigma LCP(\mathcal{R}) \leq \Sigma lcp(\mathcal{R}) \leq 2 \cdot \Sigma LCP(\mathcal{R})$.

2. Use the lcp comparison technique to modify the standard insertion sort algorithm so that it sorts strings in $\mathcal{O}(\Sigma LCP(\mathcal{R}) + n^2)$ time.

3. Give an example showing that the worst case time complexity of string binary search without precomputed lcp information is $\Omega(m \log n)$.

4. Let $S[0..n)$ be a string over an integer alphabet. Show how to build a data structure in $\mathcal{O}(n)$ time and space so that afterwards the Karp–Rabin hash function $H(S[i..j))$ for the factor $S[i..j)$ can be computed in constant time for any $0 \leq i \leq j \leq n$.

5. $\Omega(\Sigma LCP(\mathcal{R}))$ is a lower bound for string sorting for any algorithm if characters can be accessed only one at a time. However, for a small alphabet, it is possible to pack several characters into one machine word. Then multiple characters can be accessed simultaneously and treated as if they were a single *super-character*. For example, the string `abbaba` over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$ can be thought of as the string $(\texttt{ab}, \texttt{ba}, \texttt{ab})$ over the alphabet $\Sigma^2$. Algorithms taking advantage of this are called *super-alphabet* algorithms.

   Develop a super-alphabet version of MSD radix sort. What is the time complexity?