1. Describe how to modify the LSD radix sort algorithm to handle strings of varying lengths. The time complexity should be the one given in Theorem 1.14.

2. $\Omega(dp(R))$ 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 $\Sigma = \{a, b\}$ can be thought of as the string `(ab, ba, 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?

3. Use the lcp comparison technique to modify the standard insertion sort algorithm so that it sorts strings in $O(dp(R) + n^2)$ time.

4. Let $R = \{\text{manne}, \text{manu}, \text{minna}, \text{salla}, \text{saul}, \text{sauli}, \text{vihtori}\}$.
   
   (a) Give the compact trie of $R$.
   
   (b) Give the balanced compact ternary trie of $R$.

5. Show that the number of nodes in a trie $\text{trie}(R)$ is exactly $||R|| - \text{lcp}(R) + 1$, where $||R||$ is the total length of the strings in $R$ and $\text{lcp}(R)$ is as defined in Exercise 2.5. **Hint:** Consider the construction of $\text{trie}(R)$ using Algorithm 2.2.

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

7. Define

\[
MLCP[mid] = \max\{LLCP[mid], RLCP[mid]\}
\]

\[
D[mid] = \begin{cases} 
0 & \text{if } MLCP[mid] = LLCP[mid] \\
1 & \text{otherwise}
\end{cases}
\]

Show that, if we store the arrays $MLCP$ and $D$ instead of $LLCP$ and $RLCP$, we can compute $LLCP[mid]$ and $RLCP[mid]$ when needed during the string binary search.