This assignment contains a theory part and an implementation part. You should do either the theory part or the implementation part, but not both.

1 Theory Part

1. Given a Patricia tree for a collection of strings $s_1, \ldots, s_n$, show how the strings can be output in lexicographically sorted order in $O(n)$ time. More precisely, show how to construct an array $A_1, \ldots, A_n$ of pointers, where $A_i$ points to the beginning of the $i$th string in lexicographic order.

2. Let $T$ be a (not necessarily balanced) binary tree with $n$ leaves and repeatedly apply the following operations to $T$ until $T$ is of size $O(1)$.
   (a) Raking: For every maximal path $v_0, \ldots, v_k$ in $T$ that has only degree-2 vertices in its interior ($v_1, \ldots, v_{k-1}$ are all of degree 2), we delete $v_1, \ldots, v_{k-1}$ from $T$ and add the edge $v_0, \ldots, v_k$.
   (b) Pruning: Remove every leaf of $T$.

   Show that this process finishes after $O(\log n)$ iterations and that the total work done is $O(n)$. (A single iteration is easily implemented in $O(|T|)$ time, so one way to prove this is to show that $|T|$ decreases by a constant factor after each iteration.)

   Show how this process can be used in a data structure that has $O(n)$ size, takes $O(n)$ time to build and can answer lowest-common-ancestor queries in $O(\log n)$ time. (The lowest-common-ancestor of two nodes $u$ and $w$ is the node of maximum depth that has both $u$ and $w$ as descendants.)

3. Show how to use the log-size labelling scheme (where each node $v$ that is the root of a subtree of size $s(v)$ is labelled with the value $\lfloor \log_2 s(v) \rfloor$) to develop a data structure for lowest-common-ancestor queries that can be constructed in $O(n)$ time and answers queries in $O(\log n)$ time.

4. **Bonus:** Recall $x$-fast tries: They store a subset $S \subseteq \{0, \ldots, U-1\}$. Given any integer $x$, we can find the smallest value $y \in S$ such that $y \geq x$ in $O(\log \log U)$ time. Notice that, with a hash table we can test in $O(1)$ time, if $x \in S$.

   Describe a variation of an $x$-fast trie that, given any integer $x$, can find the smallest value $y \in S$ such that $y \geq x$ in $O(1 + \log \log(y - x))$ time. Your structure should not be any bigger than an $x$-fast trie, i.e., the space should be $O(n \log U)$.

2 Implementation Part

For this part of the assignment, you are to complete the implementation of the PatriciaTrie data structure provided in the ptrie.zip archive file. For full marks, your implementation should

1. Support adding (add($x$)) and removing (remove($x$)) strings from the trie. These operations should ensure that the structure remains space-efficient by ensuring that every internal node has at least 2 children.

2. Support exact searches (search($x$)), single-match prefix searches (prefixSearch1($x$), and multiple-match prefix searches (prefixSearchMany($x$)).
3. Be thoroughly tested for correctness and performance. The easiest way to do correctness (and even performance) testing is to compare the results of your structure with a TreeSet that stores strings.