1. Describe how to modify the splay-tree-based rope data structure (described in the homework handout) to support a new operation \textsc{Reverse}(S), which replaces a string $S$ with its reversal \textit{in $O(1)$ worst-case and amortized time}. The amortized times for all other operations should change by at most a small constant factor.

\textbf{Solution:} We add a single boolean flag $v\cdot \text{rev}$ to every node $v$, which indicates whether the subtree rooted at $v$ should be considered reversed. The \textsc{Reverse} algorithm is almost trivial: To reverse a string $S$, we call the following function on the root of the tree representing $S$.

\begin{verbatim}
\textsc{Reverse}(v):
  if $v \neq \text{Null}$
    $v\cdot \text{rev} \leftarrow \neg v\cdot \text{rev}
\end{verbatim}

The simplest approach to modifying the other operations is to never let them see the reversal bits. In every operation, just before we read any field of any node $v$, we run the following algorithm. (To pronounce the function name “\textsc{OkayFine}” correctly, pretend that you are a petulant teenager whose parents have been nagging you for months to clean your room.)

\begin{verbatim}
\textsc{OkayFine}(v):
  if $v\cdot \text{rev} = \text{True}$
    $v\cdot \text{rev} \leftarrow \text{False}$
    swap $v\cdot \text{left} \leftrightarrow v\cdot \text{right}$
    \textsc{Reverse}(v\cdot \text{left})
    \textsc{Reverse}(v\cdot \text{right})
\end{verbatim}

(In C++, this code could be injected transparently by overloading the $\rightarrow$ operator.) Calling \textsc{OkayFine} adds only $O(1)$ time to every node access, and therefore increases the cost of any other operation by only a constant factor.

\textbf{The fact that we’ve implemented ropes using splay trees is utterly irrelevant.}

Precisely the same lazy-propagation strategy works for any balanced binary search tree that supports \textsc{Split} and \textsc{Join} in $O(\log n)$ (possibly amortized / expected) time.

In the following figure, a node is red if and only if its reversal bit is set to \text{True}. The tree on the left is a splay-rope for the string $S = \text{STRESSED}$. The remaining steps show the execution of \textsc{Lookup}(S, 8), first performing the search and then splaying the target node up to the root.