In both problems in this homework, your goal is to design a data structure that efficiently maintains a collection of strings (sequences of characters) subject to some or all of the following operations:

- NEWSTRING(a) creates a new string of length 1 containing only the character a and returns a pointer to that string.
- CONCAT(S, T) removes the strings S and T (given by pointers) from the data structure, adds the concatenated string ST to the data structure, and returns a pointer to the new string.
- SPLIT(S, k) removes the string S (given by a pointer) from the data structure, adds the substrings $S[1..k]$ and $S[k+1..length(S)]$ to the data structure, and returns pointers to the two new strings. You can safely assume that $1 \le k \le length(S) - 1$.
- REVERSE(S) removes the string S (given by a pointer) from the data structure, adds the reversal of S to the data structure, and returns a pointer to the new string.
- LOOKUP(S, k) returns the kth character in string S (given by a pointer), or NULL if the length of S is less than k.

For example, we can build the strings `SPLAYTREE` and `UNIONFIND` with 18 calls to NEWSTRING and 16 calls to CONCAT. Further operations modify the collection of strings as follows:

operation	result	stored strings
REVERSE(`SPLAYTREE`)	`EERTYALPS`	{**`EERTYALPS`**, `UNIONFIND`}
SPLIT(`EERTYALPS`, 5)	`EERTY`, `ALPS`	{**`EERTY`**, **`ALPS`**, `UNIONFIND`}
SPLIT(`UNIONFIND`, 3)	`UNI`, `ONFIND`	{`EERTY`, `ALPS`, **`UNI`**, **`ONFIND`**}
REVERSE(`ONFIND`)	`DNIFNO`	{`EERTY`, `ALPS`, `UNI`, **`DNIFNO`**}
CONCAT(`UNI`, `EERTY`)	`UNIEERTY`	{**`UNIEERTY`**, `ALPS`, `DNIFNO`}
SPLIT(`UNIEERTY`, 5)	`UNIEE`, `RTY`	{**`UNIEE`**, **`RTY`**, `ALPS`, `DNIFNO`}
NEWSTRING(LOOKUP(`UNIEE`, 5))	`E`	{**`E`**, `UNIEE`, `RTY`, `ALPS`, `DNIFNO`}

Except for NEWSTRING and LOOKUP, these operations are destructive; at the end of the sequence above, the string `ONFIND` is no longer stored in the data structure.

In both of the following problems, you need to describe a data structure that uses $O(n)$ space, where n is the sum of the stored string lengths, describe your algorithms for the various operations, prove that your algorithms are correct, and prove that they have the necessary running times. (Most of these steps should be very easy.)

---

1. Describe a data structure that supports the following operations:
   - SPLIT, CONCAT, and LOOKUP in $O(\log n)$ time (either worst-case, expected, or amortized).
   - NEWSTRING in $O(1)$ worst-case and amortized time.
   - ***5 points extra credit:*** REVERSE in $O(1)$ worst-case and amortized time.

2. Describe a data structure that supports the following operations:
   - CONCAT in $O(\log n)$ amortized time.
   - NEWSTRING and LOOKUP in $O(1)$ worst-case and amortized time.
   - ***5 points extra credit:*** REVERSE in $O(1)$ worst-case and amortized time.

   This data structure does not need to support SPLIT at all.
