Assignment 1A

Problem Statement:

Consider an intersection with several vehicles passing by everyday. We would like to store the numberplates of all the vehicles that we see, and search for specific vehicle numbers in the future.

Intuitively, the numbers that we see on this road will look quite random. So using a Binary Search Tree to store this set will suffice. Each number looks like the strings defined in Assignment 0.

Construct, and maintain a set of such numbers using a Binary Search Tree (BST) as the data structure and support searching for a number in this set.

Input Format:

- The first line of the input will give a list of distinct numberplates $a_1, a_2, \ldots, a_n$ with each a_i separated by space. This line ends with a '\n' character.
- Each of the next lines looks like "S m \n" where m is a number plate string. (S stands for Search)
- End of input is indicated by EOF character.

Output:

No output after reading the first line. Simply create a BST with the numbers from the first line.

Once you read "S m":

- Output 1 if m exists in the list, followed by a space and a string consisting of 'L's and 'R's. The string of L's and R's is a path from root to the node containing m where 'L' and 'R' indicate left child and right child respectively. (Assume path from root to root is the empty string). End this output line with a '\n'.

- Output 0 otherwise followed by a '\n'.

Read the next input line and repeat the above.

Your program should terminate only when you read the EOF character.

Implementation Rules:

- You have to insert $a_1, a_2, \ldots, a_n$ into an empty BST in the same order.
- Searching for m in the set has to be done in the BST constructed.
- Each node of your BST should contain a pointer to the two children, and parent.
- Comparison of the elements has to be done using your own function from Assignment 0.

Remarks

- You are encouraged to use your own code from your previous assignment submissions.
- See the public test cases to understand the i/o format.
- To check that you are building your BST correctly, it might be a good idea to use the inorder traversal as a debug string. Don't forget to remove all debug string outputs before submitting your code!