CS2233: Data Structures
Assignment 8
16th October, 2018

Problem Statement

- Input: A set $S = \{a_1, a_2, \ldots, a_n\}$ of natural numbers given as a sequence. Additionally, several requests to perform operations on S.
- Goal: Store the elements of S in a Red-black tree and serve the following requests:
  1. Add an input number x into set S. (i.e., $S \leftarrow S \cup \{x\}$)
  2. Find successor of a given number x in S.
  3. Delete a given number x from S. (i.e., $S \leftarrow S \setminus \{x\}$)
  4. Search for a given number x in S.
  5. Print the set.

Input Format

Each line of the input starts with one of six symbols:

- 'N' (stands for new set)
- '+' (Add element to set)
- '>' (Find successor)
- '–' (Delete element)
- 'S' (search within most recent set)
- 'C' (Print children)
- 'P' (pre-order traversal)

The input format specification for all of the above are exactly like in assignments 2B and 3.

Output Format

- For input lines starting with 'N', 'P', '+', '–', '>' the output format is exactly as per specification in assignments 2B and 3.
- If input line was "S t":
  - Output "-1" if t is not found in the set formed by the numbers in the most recent line that started with "N".
  - Else, output a bit string that represents the path from root to the node N containing t followed by a space and the **color of** N. Represented black with 'B' and red with 'R'. End the line with a `\n` character.
- If input line was "C t": Print the value at left child followed by its color followed by the value at the right child and its color. If a child is a leaf, print 'L' instead of value.

Implementation rules

- The data structure used to implement the set should be a Red-black Tree.
- All requests are to be handled with respect to the most recent set built.

Other Remarks

- For help with testing your code, you could use the online visualizer here: `https://www.cs.usfca.edu/~galles/visualization/RedBlack.html`
- By using a BFS starting from the root of the tree, it is possible to print the entire tree layer by layer. Further, on making some rough estimates for number of space characters to use as offset, the tree can be printed on the screen to even look like a tree with the root at the horizontal center of the display. This won't work well for very large trees as you'll quickly run out of screen real estate.
- You are encouraged to reuse and modify **your own** code from previous assignments.

- It is probably a good idea to revise the data structure from lecture slides or CLRS before you begin this adventure.
- **Deadline:** Midnight of 3rd November 2018.

Grading

This assignment will deviate from the norm and count for 190 points with the following break-up:

- Insert function: 90
- Delete function: 90
- Rest (search, print, successor): 10

Note that evaluation of Insert and Delete functions require the basic procedures – search, print and successor to run correctly. Hence, if your basic procedures do not work, you will end up losing points from other parts of the break-up as well. i.e., although "Rest" counts for only 10 according to the break-up above, in reality it counts for a lot more.

Example

Input:

```
---------------------------
N 31 16
+ 45
S 45
S 31
+ 9
S 45
S 9
S 16
+ 18
+ 21
S 21
C 21
+ 24
S 21
C 31
C 21
S 18
- 21
S 21
S 18
C 18
- 31
C 18
C 24
P

---------------------------
```

The requests not mentioned in the input like `Find Successor` are exactly as in previous assignments.

Output:
Coming soon.

```
---------------------------
1 R
 B
1 B
00 R
0 B
011 R
L B L B
01 B
16 R 45 B
18 R 24 R
010 R
-1
01 B
L B 24 R
L B L B
16 R 45 B
24 16 9 18 45

---------------------------
```