BST Operations

Problem Description

You are given a sequence of commands to maintain a Binary Search Tree (BST) with unique keys.

- `insert x`: Insert integer x into the BST. The key will not be inserted if it is already present in the BST.
- `delete x`: Remove integer x from the BST. The key to be deleted is guaranteed to exist in the BST.
- `exit`: Appears exactly once as the last line. After reading it, output the level order of the BST.

Note. Deletion in a BST requires three cases:

- No child: remove directly.
- One child: replace with the child.
- Two children: replace with the inorder successor and then delete that successor.

It is important to handle these cases separately. In particular, **do not** use the inorder successor when a node has only one child; doing so will change the tree structure and your solution will not match the expected output. See Sample 2 for a concrete example.

Input Format

The input is a sequence of commands, each specifying either an insertion (e.g., `insert 3`) or a deletion (e.g., `delete 3`) in the binary search tree. Commands are executed in order and continue until the final command `exit` is encountered.

Output Format

After all operations are performed, output the elements of the binary search tree in **level order**, with **each number followed by a space character.**

Constraints

- Number of commands < 3000.

- For commands of the form `insert x` or `delete x`, the key x satisfies $1 \leq x \leq 10^9$.

Example Test Case

Sample Input 1

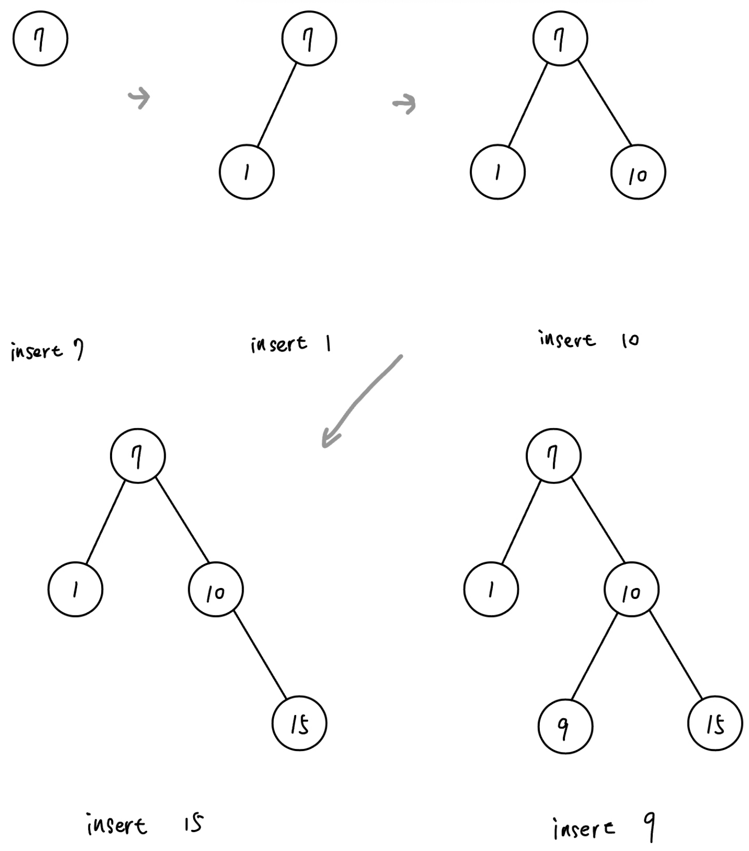
```
insert 7
insert 1
insert 10
insert 15
insert 9
exit
```

Sample Output 1

```
7 1 10 9 15
```

Explaination

This example only involves insert operations. After all insertions, the level order traversal of the BST (visiting nodes level by level from top to bottom, left to right) is 7 1 10 9 15.



Sample Input 2

```
insert 2
insert 4
insert 1
insert 8
insert 6
delete 4
exit
```

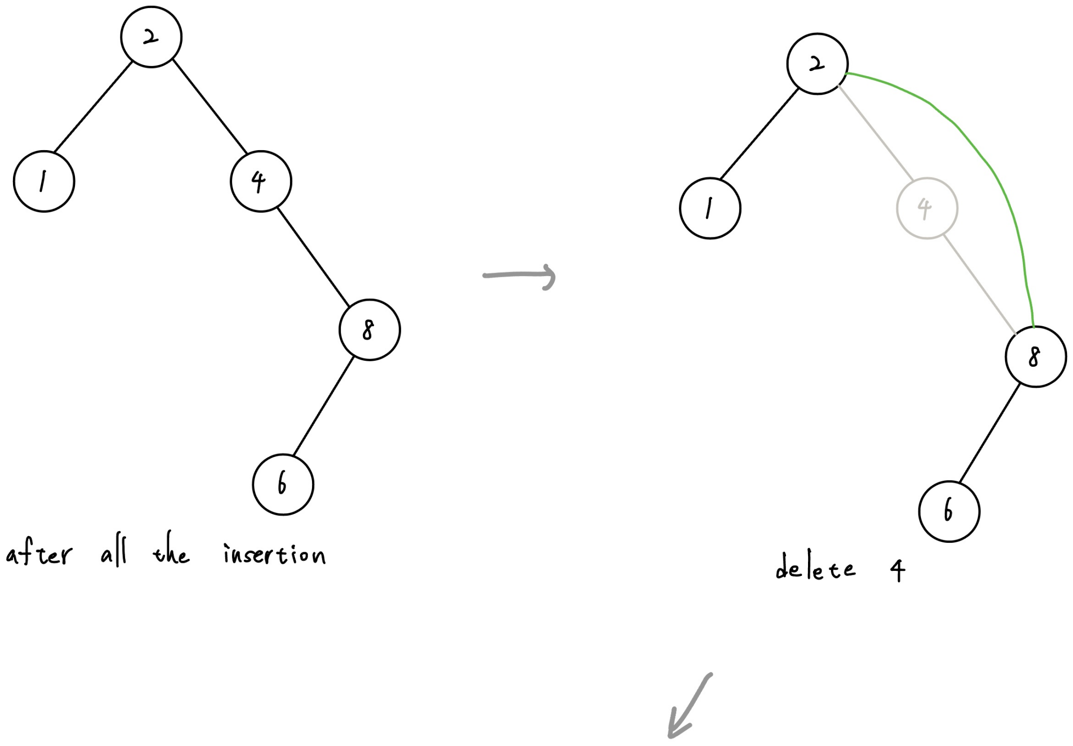
Sample Output 2

```
2 1 8 6
```

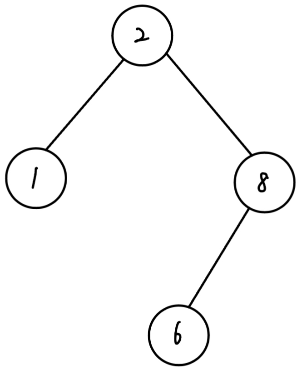
Explaination

Insert 2, 4, 1, 8, 6 in order. The BST is root 2; left child 1; right child 4 with right child 8, and 8 has left child 6.

When deleting 4, note that node 4 has only one child (the subtree rooted at 8). According to the deletion rule, we should replace 4 directly with its child. The image shows the correct result: the final BST is root 2; left child 1; right subtree rooted at 8 with left child 6. The level order is 2 1 8 6.



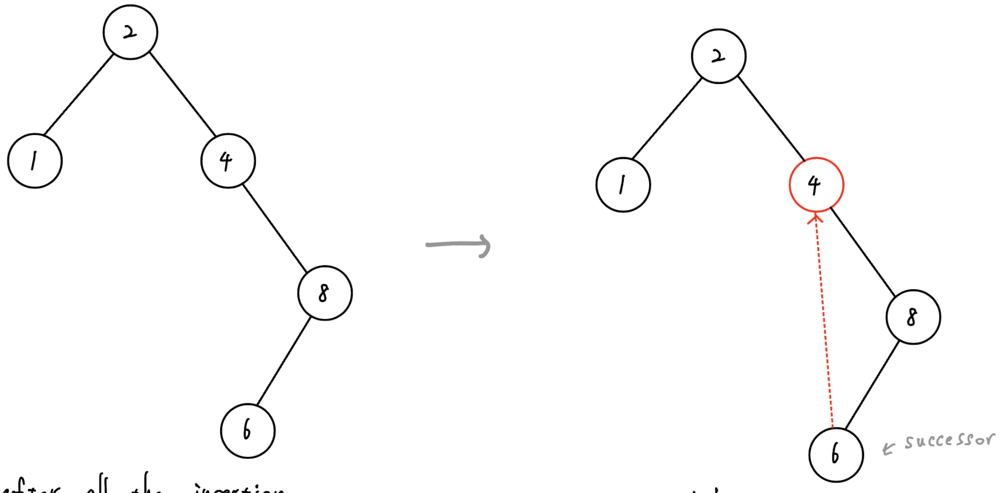
(see the next page)



result

The image below shows an alternative approach where 4 is replaced by its inorder successor 6. This is **incorrect**, because the successor rule only applies when a node has two children. Using it here changes the tree structure and does not match the expected output.

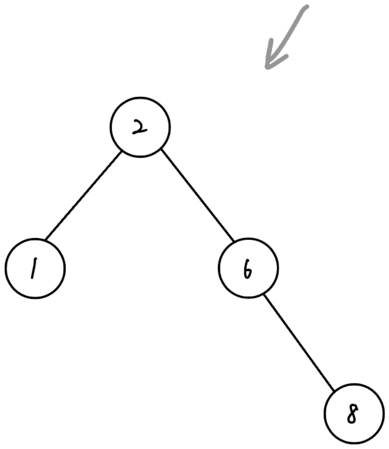
Wrong example



after all the insertion

delete 4

(see the next page)



result (wrong)

Sample Input 3

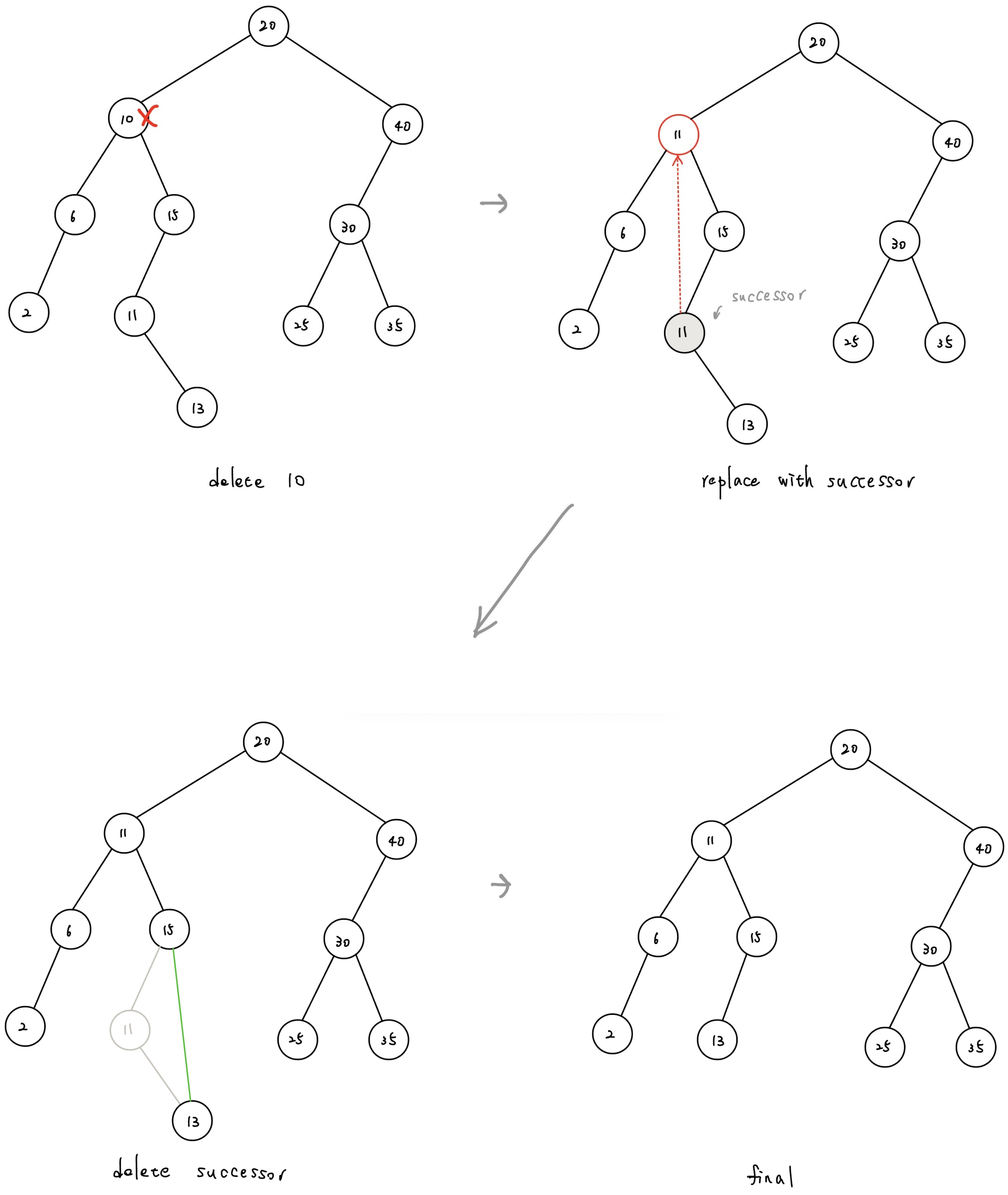
```
insert 20
insert 40
insert 30
insert 10
insert 15
insert 35
insert 25
insert 6
insert 11
insert 13
insert 2
delete 10
exit
```

Sample Output 3

```
20 11 40 6 15 30 2 13 25 35
```

Explaination

(see the next page)



delete 10

replace with successor

delete successor

final