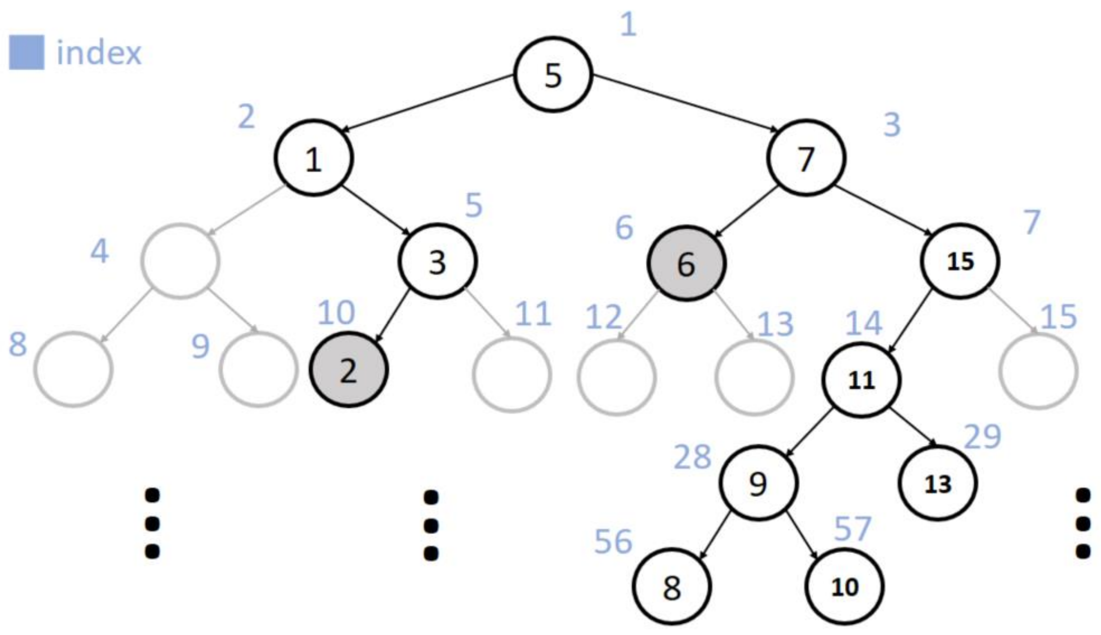
# Q10. Binary Search Tree with Linked List

Input will be n integer numbers, and you should construct these numbers into a binary search tree using linked list, with the function below:

1. Insert: insert an element (an integer) into your tree
2. search:
   - mode I : input will be the value of an element and you should output the **position** of the element.
   - mode II : input will be a position and you should output the **number** at the position
3. print: output the binary tree with sorted order.

The index (position) of your binary search tree is as below.



- the index of the node i's left child is 2i
- the index of the node i's right child is 2i+1
- the index follows the concept of complete binary search tree

## Input

Input will start with a number n, which represents the amount of the following number element. And there will be n number inputs, which you should construct them into a binary search tree. You can assume that the value of each element will not repeat.

The following line(s) will be instruction(s) for operating the tree.
Each instruction line will contain two parts, one is represented as the instruction and the other is the parameter(s) of the instruction.
The instruction will be character 'i', 'sp', 'sn' or 'p':
'i' is insert, 'sp' for searching position, 'sn' for searching element value and 'p' is print.

For example:
i 5
⇒ insert number 5 into your tree

sp 3
⇒ search the position of 3 (if there is no 3 in your tree, please print “3 is not in the tree”)
⇒ return the position number

sn 15
⇒ traverse your tree to the position 15, and print out the number at the position (Suppose there must be a node at this position)
⇒ return value of an element (a number)

p
⇒ output the whole tree with the sorted order

Please implement the file I/O part.
Read file to get input.
The file name should be “input.txt”.

## Sample Input (input.txt)

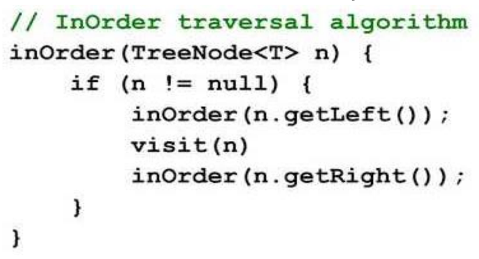
10
5 7 1 3 15 11 9 13 8 10
i 6
i 2
sp 15
sn 6
p

## Sample Output

7
6
1 2 3 5 6 7 8 9 10 11 13 15

## Hint

In-Order Traversal:
You can use this traversal to print an sorted order in a binary search tree.



```
// InOrder traversal algorithm
inOrder(TreeNode<T> n) {
    if (n != null) {
        inOrder(n.getLeft());
        visit(n)
        inOrder(n.getRight());
    }
}
```