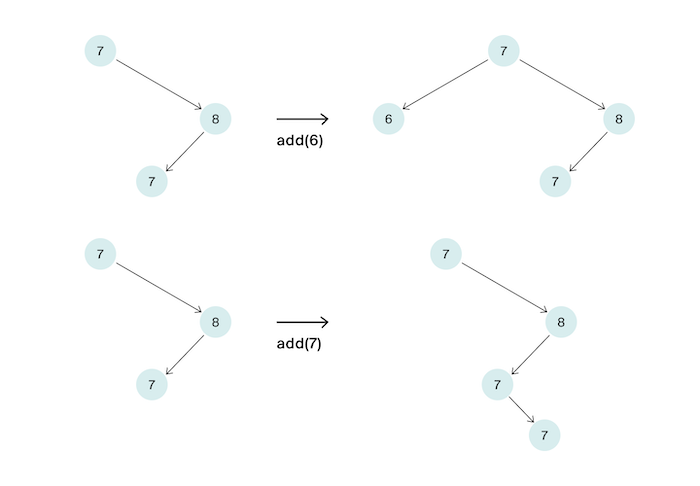
J. Добавь узел

Дано BST. Надо вставить узел с заданным ключом. Ключи в дереве могут повторяться.

На вход функции подаётся корень корректного бинарного дерева поиска и ключ, который надо вставить в дерево. Осуществите вставку этого ключа. Если ключ уже есть в дереве, то его дубликаты уходят в правого сына. Таким образом вид дерева после вставки определяется однозначно. Функция должна вернуть корень дерева после вставки вершины.

Ваше решение должно работать за O(h), где h –— высота дерева.

На рисунках ниже даны два примера вставки вершин в дерево.



Формат ввода

Ключи дерева – натуральные числа, не превосходящие 109. Число вершин в дереве не превосходит 105.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение. Ниже приведены сигнатуры функций, которые надо реализовать.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/lVil2dE14f0nUg).

**Python**

*# do not declare Node in your submit-file*   
**class** Node:   
    **def** \_\_init\_\_(self, left=None, right=None, value=0):   
        self.value = value   
        self.right = right   
        self.left = left   
   
**from** node **import** Node *# Attention!*   
**def** insert(root: Node, key: **int**) -> Node

**C++**

*// do not declare Node in your submit-file*   
**struct** Node {   
    Node∗ left;   
    Node∗ right;   
    **int** value;   
};   
**#include** "solution.h" *// Attention!*   
Node∗ insert(Node∗ root, **int** key);

**Java**

*// do not declare Node in your submit-file*   
**public** **class** Node {   
    **private** Node left;   
    **private** Node right;   
    **private** **int** value;   
    **public** **int** getValue();   
    **public** Node getRight();   
    **public** Node getLeft();   
    **public** **void** setValue(**int** value);   
    **public** **void** setRight(Node right);   
    **public** **void** setLeft(Node left);   
    **public** Node(Node left, Node right, **int** value);   
}   
**public** **class** Solution {   
        **public** **static** Node insert(Node root, **int** key);   
}

**Go**

*// do not declare Node in your submit-file*   
type Node **struct** {   
        value    **int**   
        left   ∗Node   
        right  ∗Node   
}   
package main   
   
func insert(root ∗Node, key **int**) ∗Node

**NodeJs**

*// do not declare Node in your submit-file*   
**class** Node {   
    constructor(value) {   
        **this**.value = value;   
        **this**.left = **null**;   
        **this**.right = **null**;   
    }   
}   
function insert(node, key) {   
*// Your code*   
}