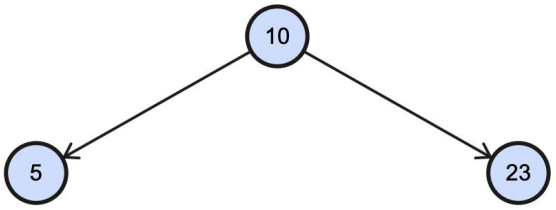
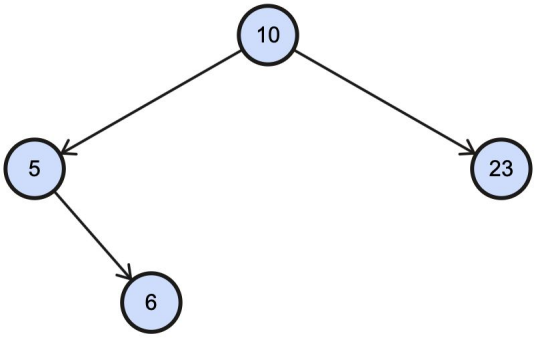
Доп. статья:
Статья про двоичные деревья:
[Алгоритмы и структуры данных для начинающих: двоичное дерево поиска]

Задача:

Перед вами простейшее двоичное (иногда называют бинарное) дерево поиска. У этой структуры данных есть узлы (кружки с числами) и ребра (линии, соединяющие кружки). Узел с числом 10 называется родителем, а узлы 5 и 23 называются детьми (5 - левый сын, 23 - правый сын).



Основное свойство двоичного дерева поиска заключается в том, что значение в правом сыне всегда больше или равно значения в родительском узле, а значение в левом сыне всегда меньше родительского. Действительно, мы видим на примере выше, что правый сын содержит значение 23 (больше, чем у родителя), а левый сын содержит значение 5 (меньше, чем у родителя). Поэтому, перед нами корректное двоичное дерево поиска.
Мы можем добавлять сколько угодно значений в двоичное дерево поиска. Если мы захотим добавить в наше дерево новое значение 6, дерево изменится таким образом:



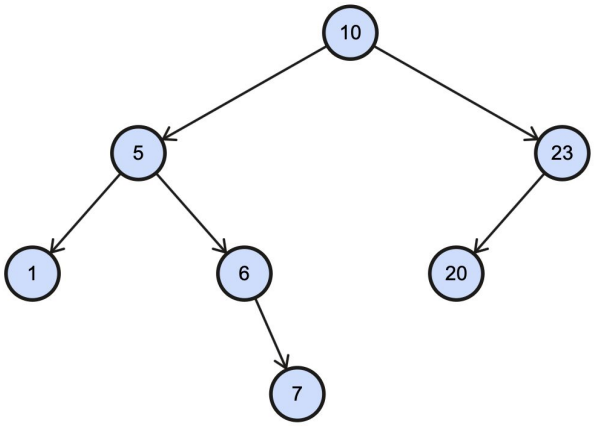
Логика добавления следующая:

- 6 больше или меньше 10? -> меньше, поэтому должно уйти в левое поддерево
- 6 больше или меньше 5? -> больше, поэтому должно уйти в правое поддерево

В результате такого добавления, наше двоичное дерево осталось корректным. Для всех родителей и сыновей сохраняется свойство, описанное выше (левый сын всегда меньше родителя, правый сын больше или равен родителю).

Теперь в нашем дереве два родителя - узел 10 (родитель для узлов 5 и 23) и узел 5 (родитель для узла 6).

Давайте добавим еще три значения в наше двоичное дерево поиска. Добавим 1, 20 и 7.
Получим такое дерево:



Каждое из значений добавлялось по логике, описанной выше:

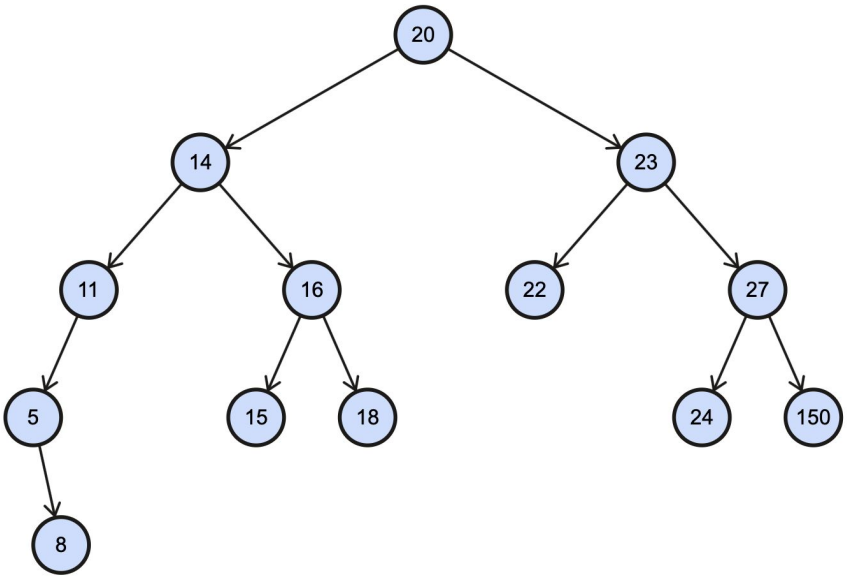
```
1 больше, чем 10? -> нет -> уходим влево
1 больше, чем 5? -> нет -> уходим влево

20 больше, чем 10? -> да -> уходим вправо
20 больше, чем 23? -> нет -> уходим влево

7 больше, чем 10? -> нет -> уходим влево
7 больше, чем 5? -> да -> уходим вправо
7 больше, чем 6? -> да -> уходим вправо
```

Вы можете потренироваться в добавлении значений в двоичное дерево поиска на этом сайте: http://btv.melezinek.cz/binary-search-tree.html

Теперь, в нашем дереве 4 родителя - узел 10 (для 5 и 23), узел 5 (для 1 и 6), узел 23 (для 20) и узел 6 (для 7).

Так как для всех родителей в этом дереве сохраняется основной закон двоичного дерева поиска (левый сын меньше, правый сын больше или равен), это двоичное дерево поиска корректное.

Интересное свойство двоичного дерева поиска заключается в том, что поддеревья тоже являются двоичными деревьями поиска.



Вам необходимо создать двоичное дерево поиска, изображенное на картинке выше.

Для этого создайте класс “Узел” (англ. `Node`), который будет содержать поля-ссылки на два других узла (левый и правый сын).
Затем, создайте корневой (англ. `root`) узел (со значением 20).
После этого, необходимо реализовать метод, который будет добавлять новые узлы в ваше дерево.
Этот метод должен принимать в качестве аргументов добавляемое значение и ссылку на корень дерева. Проходясь по дереву, он должен вставлять новый узел в правильное место дерева.

Когда двоичное дерево, изображенное выше, будет создано, необходимо используя рекурсию вывести в консоль все числа из этого двоичного дерева поиска в отсортированном виде. Ваше решение должно работать для любого корректного двоичного дерева поиска.
Этому методу необходимо передавать на вход ссылку на корень дерева.