1. **Convert BST to Min Heap**

Given a binary search tree which is also a complete binary tree. The problem is to convert the given BST into a Min Heap with the condition that all the values in the left subtree of a node should be less than all the values in the right subtree of the node. This condition is applied on all the nodes in the so converted Min Heap.

**Examples:**

Input : 4

/ \

2 6

/ \ / \

1 3 5 7

Output : 1

/ \

2 5

/ \ / \

3 4 6 7

The given **BST** has been transformed into a

**Min Heap.**

All the nodes in the Min Heap satisfies the given

condition, that is, values in the left subtree of

a node should be less than the values in the right

subtree of the node.

# Merge two binary Max Heaps

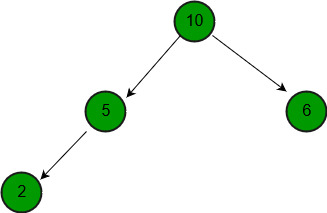
Given two binary max heaps as arrays, merge the given heaps.

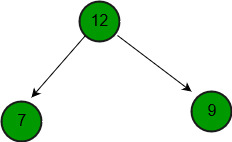
**Examples :**

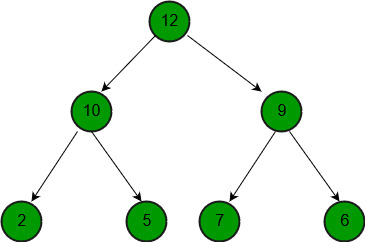
Input : a = {10, 5, 6, 2},

b = {12, 7, 9}

Output : {12, 10, 9, 2, 5, 7, 6}







1. **Binomial Heap is an extension of** [**Binary Heap**](http://geeksquiz.com/binary-heap/) **that provides faster union or merge operation together with other operations provided by Binary Heap.**

*A Binomial Heap is a collection of Binomial Trees*

**What is a Binomial Tree?**  
A Binomial Tree of order 0 has 1 node. A Binomial Tree of order k can be constructed by taking two binomial trees of order k-1 and making one as leftmost child or other.  
A Binomial Tree of order k has following properties.  
a) It has exactly 2k nodes.  
b) It has depth as k.  
c) There are exactly kCi nodes at depth i for i = 0, 1, . . . , k.  
d) The root has degree k and children of root are themselves Binomial Trees with order k-1, k-2,.. 0 from left to right

**Binomial Heap:**  
A Binomial Heap is a set of Binomial Trees where each Binomial Tree follows Min Heap property. And there can be at most one Binomial Tree of any degree.

**Examples Binomial Heap:**

12------------10--------------------20

/ \ / | \

15 50 70 50 40

| / | |

30 80 85 65

|

100

A Binomial Heap with 13 nodes. It is a collection of 3

Binomial Trees of orders 0, 2 and 3 from left to right.

10--------------------20

/ \ / | \

15 50 70 50 40

| / | |

30 80 85 65

|

100

**Operations of Binomial Heap:**  
The main operation in Binomial Heap is union(), all other operations mainly use this operation. The union() operation is to combine two Binomial Heaps into one. Let us first discuss other operations, we will discuss union later.

**1)** insert(H, k): Inserts a key ‘k’ to Binomial Heap ‘H’. This operation first creates a Binomial Heap with single key ‘k’, then calls union on H and the new Binomial heap.

**2)** getMin(H): A simple way to getMin() is to traverse the list of root of Binomial Trees and return the minimum key. This implementation requires O(Logn) time. It can be optimized to O(1) by maintaining a pointer to minimum key root.

**3)** extractMin(H): This operation also uses union(). We first call getMin() to find the minimum key Binomial Tree, then we remove the node and create a new Binomial Heap by connecting all subtrees of the removed minimum node. Finally, we call union() on H and the newly created Binomial Heap. This operation requires O(Logn) time.

**4)** delete(H): Like Binary Heap, delete operation first reduces the key to minus infinite, then calls extractMin().

**5)** decreaseKey(H): decreaseKey() is also similar to Binary Heap. We compare the decreases key with it parent and if parent’s key is more, we swap keys and recur for the parent. We stop when we either reach a node whose parent has a smaller key or we hit the root node. Time complexity of decreaseKey() is O(Logn).