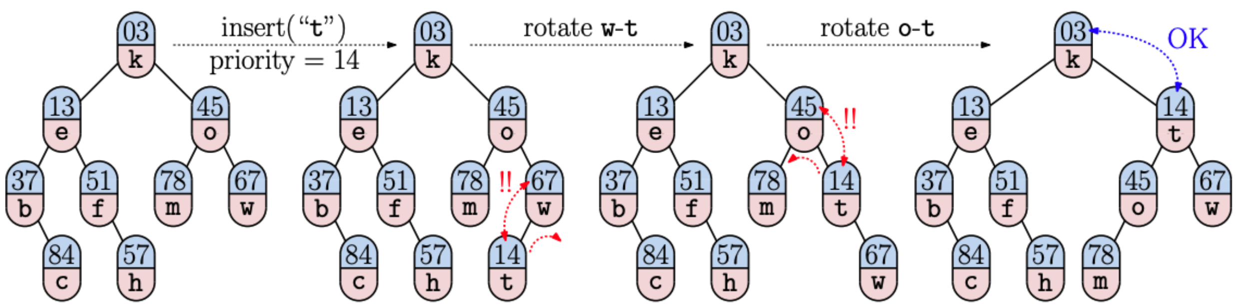
Indian Institute of Technology, Tirupati
March 28th, 2023

CS204L : Data Structures & Algorithms
Mid - Semester Examination

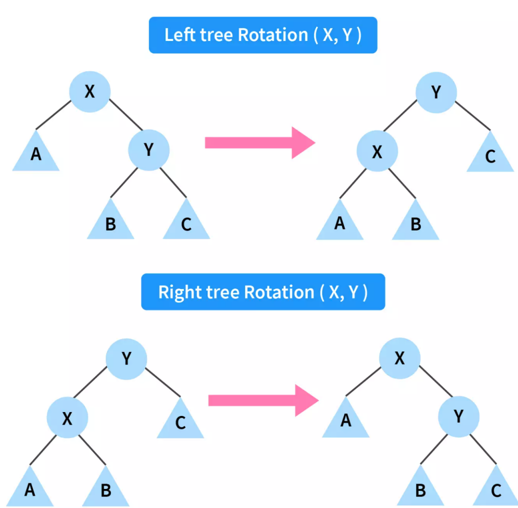
Select the problem based on the number assigned to you. [100 Marks]

1. Implement a Binary Search Tree with each node has a pair **(key, priority)**, where we maintain the BST property with respect to the *key*, *i.e.*, for a node q, all its left children have *key* values less than the *key* value of the node and all the right children have their *key* values greater than or equal to the *key* value of the node, *i.e.*, **x.key** $<$ **q.key** $\leq$ **y.key,** $x \in L_q, y \in R_q$. This being recursively true for all nodes implies this condition must be satisfied by all nodes of BST.
   Also, for each node q, a MAX-HEAP property is maintained with respect to the *priority* values, *i.e.*, the *priority* value of the node q is greatest among him and all its childrens' *priority* values, *i.e.*, **(q.left.priority** $<$ **q.priority** $>$ **q.right.priority)** which is recursively true for all nodes, implies that this condition must be satisfied by all nodes of BST.

   Implement: **insert(Node(***key*,*priority***))** where we insert the node into the tree. Do a normal BST insert. Then you need to use left rotation or right rotation to maintain heap property. Example:



   Here the key *key* values are in pink and *priority* values are in blue. Whenever the *priority* value property is broken, do a left or right rotation; **in the example, it follows a min-heap property but implements it as a max-heap property in the actual problem.** [Safely assume all *key* values and *priority* values are distinct]



   Implement: **extractMaxpriority(Node *root)** which is similar to a heap-extract-max except here you need to delete the node with max priority and rearrange the tree such that all other properties are satisfied.

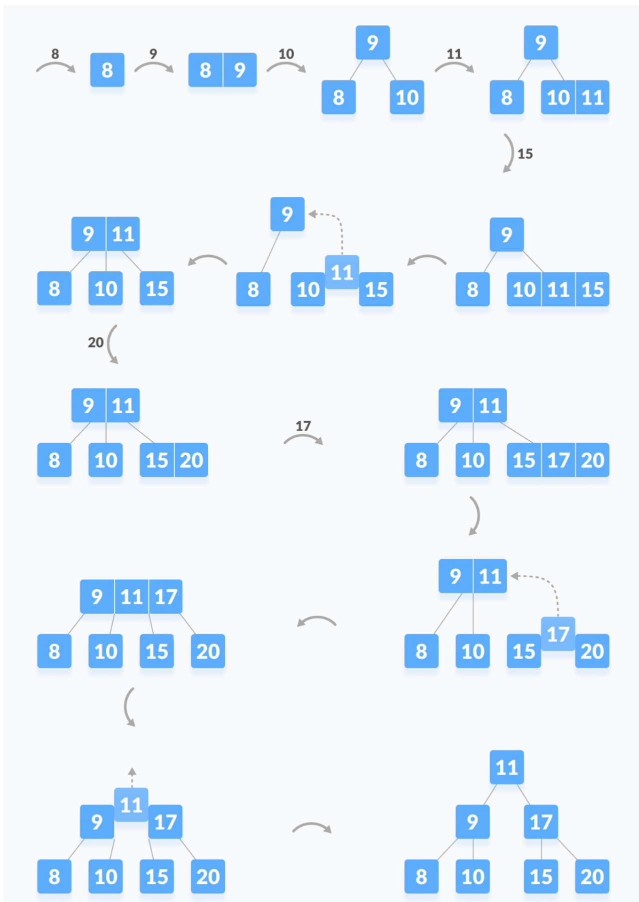
Hint: Do a normal BST delete using the successor node. Now remove the successor node and do rotation on the node (downwards if required) where the successor values are installed.

2. Implement a generalised Search Tree that can have more than one key element for each node. So a generalised search tree of order **"m"** must satisfy:

   (a) Every node has at most m children and $m - 1$ keys.

   (b) Every item inside the node is kept in non decreasing order.

   For example, if we need to insert the following keys: 8, 9, 10, 11, 15, 20, 17 in a tree of order, m = 3.



   For selecting the element in a node that is to be shifted in the upper level you can use **median().** [Assume all keys are distinct]

You can choose to solve the problem below if you find yourself unable to solve the problem above [60 marks]

Implement a K-ary Max Heap with insertion and Extract Max operations.