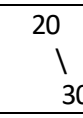
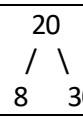
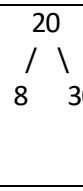
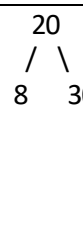
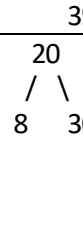
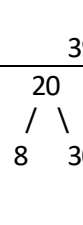
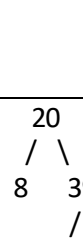
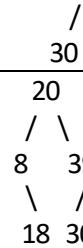
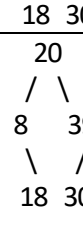
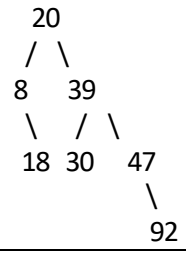
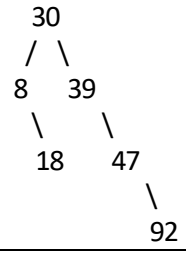
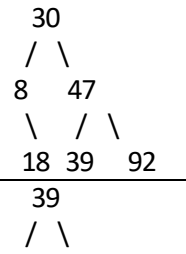
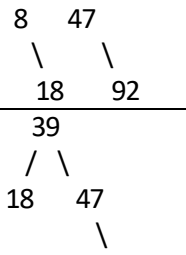
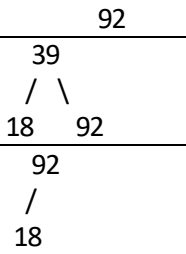
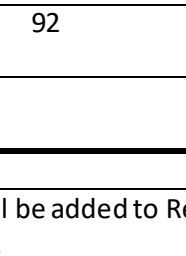
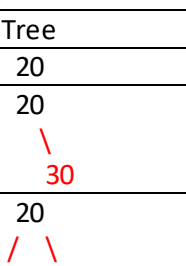
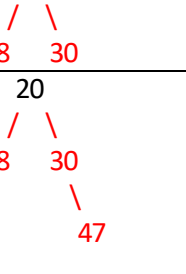
AVL Tree

20 30 8 47 39 18 20 92 will be added to AVL Tree Search Tree Data Structure respectively and remove them one by one respectively again.

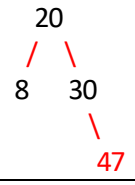
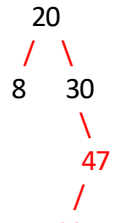
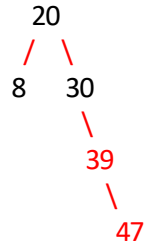
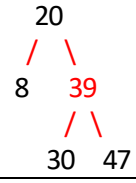
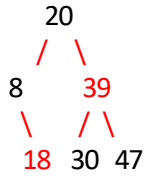
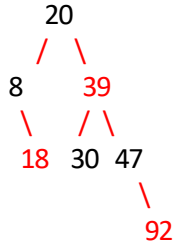
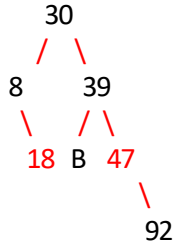
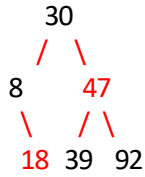
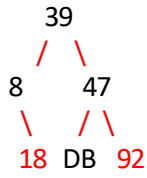
Tree	Explenation
	20 added to AVL Tree as root node
	Because 30 bigger than 20, 30 added right child (right height = 1, left height = 0)
	Because 8 less than 20, 8 added left child (right height = 1, left height = 1)
	Because 47 bigger than node 20 and 30, 47 will be added right child of node 30 (right height = 2, left height = 1)
	Because 39 bigger than 20 and 30, 39 will be added left child of node 47 (right height = 3, left height = 1)
	Because this tree is RightLeft subtree, we have to rotate this subtree right left.
	Now we have to rotate this tree left
	Now we have right height = 2, left height = 1
	Because 18 less than 20 and bigger than 8, 18 added right child of node 8 (right height = 2, left height = 2)
	20 will not be added to tree because there is node with data 20.

	Because 92 bigger than 20,39 and 47, 92 added right child of node 47 (right height = 3, left height = 2)
	When remove 20, it has 2 child node, we replace node that we delete with its successor which is 30 in this problem. After than, we have to make this tree that follows AVL tree rules. There is RR subtree there so we have to figüre this tree out (right height = 3, left height = 2)
	There is left rotation there and now tree is balanced(right height = 2, left height = 2)
	When remove 30, it has 2 child node, we replace node that we delete with its successor which is 39 in this problem. (right height = 2, left height = 2)
	When we remove 8, because it has 1 child, we can replace with its child (right height = 2, left height = 1)
	When we remove 47, because it has 1 child, we can replace with its child (right height = 1, left height = 1)
	When remove 39, it has 2 child node, we replace node that we delete with its successor which is 92 in this problem. (right height = 0, left height = 1)
	When we remote 18, because it has no child, we can remove it directly
	When we remote 92, because it has no child, we can remove it directly

Red-Black Tree

20 30 8 47 39 18 20 92 will be added to Red-Black Search Tree Data Structure respectively and remove them one by one respectively again.

Tree	Explenation
	20 added to AVL Tree as root node and color it as red
	30 added as red first then check if parent is root node. Because 20 is root node, we left tree as it is.
	8 added as red first then check if parent is root node. Because 20 is root node, we left tree as it is.
	47 added as red first then check if parent is root node. Because 30 is not root node, we will look at sibling of its parent. Because color of node 47's parent's sibling is red which is 8, we recolor

	Parent's parent is 20 which is root node so we do not recolor it
	39 added as red first then check if parent is root node. Because 47's color is red,we look for parent's siblings. Sibling of 47 is null so we do rotation. We have RL tree so we will do right left rotations respectively
	First,we rotated to right. Now we have to rotate to left.
	Second rotation is done and recolouring is also done
	18 added. Parent of 18 is black so we left this as it is
	18 added. Parent of 18 is black so we left this as it is
	For deleting 20, because 20 is black, we look at its in order successor which is 30 then replace 20 with its successor. Now 20 has no child so we delete 20 directly and replace it with double black. Then we make its parent black and its sibling red and childeren of 92 is black. Then we have to rotate this subtree to left
	When we do the operations above, we obtain this kind of tree.
	When we delete 30, because 30 is black, we look at its in order successor which is 39 then replace 30 with its successor. Now 30 has no child so we delete 30 directly and replace it with double black. Then we make its parent black and its sibling red. Then we delete DB node

39 / \ 18 47 \ 92	We can delete 8 easily because it has 1 children and its children is red which is 18 so we can replace 18 by node 8 without changing color
39 / \ 18 92	We can delete 47 easily because it has 1 children and its children is red which is 92 so we can replace 92 by node 47 without changing color
92 / 18	When we delete 39, because 39 is black, we look at its in order successor which is 92 then replace 39 with its successor. Now 39 has no child so we delete 30 directly and replace it with double black. Its parent is root so it was red so we didn't have to change its color but we have to change of its sibling's color to red
92	Because 18 is red and it has no childeren, we can delete it easily
NULL	END

2-3 Tree

20 30 8 47 39 18 20 92 will be added to 2-3 Tree Search Tree Data Structure respectively and remove them one by one respectively again.

Tree	Explenation
20	We created a new 2-node that contains the new item
20,30	Because node can have 2 node, we add 30 as second node.
8,20,30	We added 8 like this because 8 is smaller than 20 but node cannot have 3 values so we split this tree
20 / \ 8 30	We splitted our 2-3 tree data structure into 1 node and two children.
20 / \ 8 30,47	Because 47 is greater than 20 and 30 and a node can have 2 values, we can make our tree like shown.
20 / \ 8 30,39,47	We added 39 like this because 39 is greater than 20 but node cannot have 3 values so we split this tree
20,39 / | \ 8 30 47	We splitted our 2-3 tree data structure into 2 node and three child.
20,39 / | \ 8,18 30 47	We added 18 like this because 18 is less than 20 but more than 8.
20,39 / | \ 8,18 30 47	20 is already in the tree
20,39 / | \ 8,18 30 47,92	Because 92 is either greater than 39 and 47, we add it as second value of node that include 47
18,39 / | \ 8 30 47,92	We delete 20 by swapping element that is closest to 20 which is 18.
18 / | \ 8 39 47,92	We delete 30 by swapping element that is closest to 30 which is 39. But this tree doesn't satisfy requirements so we have to figure this out.

18,47 / | \ 8 39 92	We found parent's successor and add it to parent as second value.
47 / | \ 18 39 92	We swapped 8 and 18 which is closest element to 8 and delete it but it doesn't satisfy requirements so we have to figure this out.
18,47 / \ 39 92	We found parent's successor and add it to parent as second value.
18 / \ 39 92	We delete 47 and there is no need to operation over tree
/ \ 18 92	We swapped 39 with its parent's successor and delete 39 but there is no value at parent and there is two element at childs so we add it to parent's values
18,92	We add it to parent's values
92	We delete 18
NULL	We delete 92

Skip List

20 30 8 47 39 18 20 92 will be added to Skip List Data Structure respectively and remove them one by one respectively again.

List	Explanation
S0: -∞ -> 20 -> +∞	We added 20 to the first level of our skip list
S1: -∞ -> -> 30 -> +∞ S0: -∞ -> 20 -> 30 ->+∞	We added 30 by finding 30 at the first level and add next to it. Number of the element at the bottom is become 2 so we have to increase level of lists S by one and add last element untill see the
S1: -∞-> -> -> 30 -> +∞ S0: -∞ ->8 -> 20 -> 30 ->+∞	When adding 8, we start from highest level which is S2 and see next node. Next node has value 30 so go 1 level below. See first element which is 20. 8 is less than 20 and there is no below level so insert head of level 1.
S2: -∞ -> -> -> -> 47 -> +∞ S1: -∞-> -> -> 30 -> 47 -> +∞ S0: -∞-> 8 ->20->30 -> 47 -> +∞	We added 47 by finding 30 at the first level and add next to it. Number of the element at the bottom is become 4 so we have to increase level of lists S2 by one and add last element untill see the
S2: -∞ -> -> -> -> -> 47 -> +∞ S1: -∞-> -> -> 30 -> -> 47 -> +∞ S0: -∞-> 8 ->20->30 -> 39 -> 47 -> +∞	We added 39 by finding 30. Then pass one level below. 47 is less than 39 and there is no level below so we add 39 there.
S2: -∞-> -> -> -> -> -> 47 -> +∞ S1: -∞-> -> 18 -> ->30 -> -> 47 -> +∞ S0: -∞-> 8 -> 18 ->20->30 -> 39 -> 47 -> +∞	We added 18 by looking top level. Next of -∞ at S3 is +∞. 18 is less than +∞ so we pas sone level below. Next of -∞ at S2 level is 30. 18 is less than 30 so we pass one level below again. We see 8 and 18 is greater than 8 and less than 20 so we add 18 between these two nodes. 18 is a multiple of 2 so we can add one node above to node at bottom 18.
S2: -∞-> -> -> -> -> -> 47 -> +∞ S1: -∞-> -> 18 -> ->30 -> -> 47 -> +∞ S0: -∞-> 8 ->18 ->20 ->30 -> 39 -> 47 -> +∞	We want to add 20 by looking top level. Next of -∞ at S3 level is +∞. 20 is less than +∞ so we pass one level below. Next of -∞ at S2 level is 30. 20 is less than 30 so we pass one level below again. We go over nodes at level S0 and we see that 20 already at the list.

S2: -∞-> -> -> -> -> -> 47 -> ->+∞ S1: -∞-> ->18 -> ->30 -> -> 47 -> -> +∞ S0: -∞-> 8 ->18 ->20 ->30 -> 39 -> 47 -> 92-> +∞	We want to add 92 by looking top level. Next of -∞ at S3 level is +∞. 92 is less than +∞ so we pass one level below. Next of -∞ at S2 level is 30. 92 is greater than 30 so we pass node 30. Then because next node of 30 is +∞, we go one level bottom. Then because there is no level below S0, we iterate over level S0 and add 92 to end of the level 92
S2: -∞-> -> -> -> -> 47 -> ->+∞ S1: -∞-> ->18 ->30 -> -> 47 -> -> +∞ S0: -∞-> 8 ->18 ->30 -> 39 -> 47 -> 92-> +∞	We want to find 20. We do the same finding method and we find 20 then we delete it. After then we look that if there is any node below it. There was no node below 20 so we end deletion.
S2: -∞-> -> -> -> 47 -> ->+∞ S1: -∞-> -> 18 -> -> 47 -> -> +∞ S0: -∞-> 8 -> 18 -> 39 -> 47 -> 92-> +∞	We want to find 30. We do the same finding method and we find 20 at S2 level. Then we delete all nodes until we reached S0 level so we end deletion.
S2: -∞-> -> -> 47 -> ->+∞ S1: -∞-> 18 -> -> 47 -> -> +∞ S0: -∞-> 18 -> 39 -> 47 -> 92-> +∞	We want to find 8. We do the same finding method and we find 8 then we delete it. After then we look that if there is any node below it. There was no node below 8 so we end deletion.
S1: -∞-> 18 -> -> -> +∞ S0: -∞-> 18 -> 39 -> 92-> +∞	We want to find 47. We do the same finding method and we find 47 at S2 level. Then we delete all nodes until we reached S0 level so we end deletion. Number of node at the level S0 become 3 which is lower than 2^2 so we can delete level S2
S1: -∞-> 18 -> -> +∞ S0: -∞-> 18 -> 92-> +∞	We want to find 39. We do the same finding method and we find 39 then we delete it. After then we look that if there is any node below it. There was no node below 39 so we end deletion.
S0: -∞-> 92-> +∞	We want to find 18. We do the same finding method and we find 18 at S1 level. Then we delete all nodes until we reached S0 level so we end deletion. Number of node at the level S0 become 1 which is lower than 2^1 so we can delete level S1
	We want to find 92. We do the same finding method and we find 92 then we delete it. After then we look that if there is any node below it. There was no node below 92 so we end deletion. Number of node at the level S0 become 0 which is lower than 2^0 so we can delete level S0.

B-Tree with order 4

20 30 8 47 39 18 20 92 will be added to T-Tree with order 4 Search Tree Data Structure respectively and remove them one by one respectively again.

Tree	Explenation
20, ,	20 added to root array
20, 30,	30 added to root array
8, 20, 30	8 added to array by shifting 20 and 30 by one right
30, , / \ 8,20, 47, ,	47 is bigger than 30 so we have to add root array but B-Tree is order 4 so we have maximum 3 element in array so we make our tree right bias and take 30 as root node and split 47 and 8,20 into two arrays
30, , / \ 8,20, 39,47,	39 is bigger than 30 so we move right child and because capacity is not full, we add 39 to that array
30, , / \ 8,18,20 39,47,	18 is less than 30 so we move left child and because capacity is not full, we add 18 to that array

30, , / \ 8,18,20 39,47,	20 is already in tree so we don't add 20 to tree
30, , / \ 8,18,20 39,47,92	92 is bigger than 30 so we move right child and because capacity is not full, we add 92 to that array
30, , / \ 8,18, 39,47,92	Because size of array includes 20 is 2 after deletion, we don't have to get number from siblings.
18, , / \ 8, , 39,47,92	18 is left most position so we take element from left subtree but now left subtree of root node has 1 element which is less than minimum number of element that children have have to do one more operation.
39, , / \ 8,18 , 47,92,	We move smallest element from right sibling of array contains 8 which is 18 to parent node and move element at parent to left child
39, , / \ 18 , , 92, ,	We delete 47 but we have a problem. We have less number of element at our array that contains 92 and we have not got enough element to take from our sibling so we merge our arrays together.
18,39,92	We obtain this kind of array
18,92	39 deleted from array
92	18 deleted from array
NULL	92 deleted from array