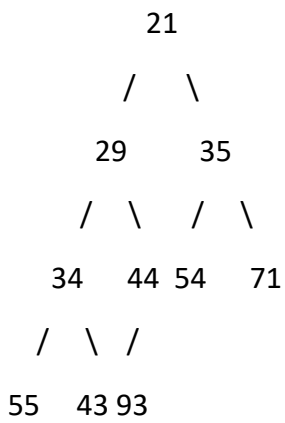
Min Heapify Function to Build Min Heap CodeStudio

You are given an array 'ARR' of integers having 'N' elements. Your task is to convert the input array into a min-Binary Heap.

A min-Binary heap is a complete binary tree in which the value of each internal node is smaller than or equal to the values of the children of that node.

Example: [34, 43, 54, 21, 44, 35, 71, 55, 29, 93]

Output: [21, 29, 35, 34, 44, 54, 71, 55, 43, 93]



Approach 1: Function to build a min heap iteratively

- **Function Purpose:** To build a min heap from an array using iterative min-heapify.
- **Explanation:**
    - Start from the last non-leaf node and move up to the root.
    - At each step, apply minHeapifyIterative to adjust the heap.
    - The iterative approach is efficient in terms of space complexity as it operates in-place.
- **Time Complexity: O(N * log(N)), where N is the number of elements in the array.**
- **Space Complexity: O(1) since it operates in-place.**

Approach 2: Function to build a min heap recursively

- **Function Purpose:** To build a min heap from an array using recursive min-heapify.
- **Explanation:**
    - Start from the last non-leaf node and move up to the root.
    - At each step, apply minHeapifyRecursive to adjust the heap.

    - The recursive approach provides a more natural understanding of the algorithm but has higher space complexity due to the call stack.
- **Time Complexity: O(N * log(N)), where N is the number of elements in the array.**
- **Space Complexity: O(log(N)) due to the call stack.**

Approach 3: Function to build a min heap using STL Priority Queue (Min Heap)

- **Function Purpose:** To build a min heap from an array using a priority queue (STL).
- **Explanation:**
    - Create a min-heap (priority queue with **greater** comparison) and push all elements into it.
    - Extract elements from the priority queue to obtain the min heap.
    - This approach is straightforward but less efficient in terms of time complexity.
- **Time Complexity: O(N * log(N)) to insert elements into the priority queue.**
- **Space Complexity: O(N) due to the priority queue.**

Conclusion:

- Both the iterative and recursive approaches have the same time complexity (O(N * log(N)).
- The iterative approach is more space-efficient with O(1) space complexity.