(22) k^{th} largest in binary tree.

- create min heap with priority queue.
- iterate queue to return k^{th} element.

```
KLargest (Node n, int k){
    PriorityQueue q = new PriorityQueue (int v
        (Node n1, int a, Node n2) => { return b - a; });
    TreeToHeap (n, q);
    int i = 0; Node r = null;
    while (i < k) {
        r = q.poll();
        i++;
    }
    return r;
}

TreeToHeap (Node n, PriorityQueue <Node> q) {
    if (n == null) return;
    q.offer (n.data);
    TreeToHeap (n.left);
    TreeToHeap (n.right);
}
```

Better Approach create max heap and keep checking the size of heap. if size == k, don't add element until thy are less than heap.peek(). maintain size = k for heap. This saves space.