

while Is empty (1)? Node temp = S. tap(1; s. pap(1); ans. push back (temp > data); if (temp > night) s. push (temp > night); if (temp > left) s. push (temp > left); ? Noturn ahs; ? Post ander: 1 8 4 5 2 6 7 3 1 (LRN) I We know that we have print pades in the ander — LRN.	
Node *temp = S. tap(): s. pap(): ans. push back (temp \right); if (temp \right) right); if (temp \right) left); s. push (temp \right); 3 return ans; 3 * Rost Orden Traversal; Post anden: 1 8 4 5 2 6 7 3 1 (LRN) be know that we have print nodes in the ander \right, RN.	
Node *temp = S. tap(): s. pap(): ans. push back (temp \right); if (temp \right) right); if (temp \right) left); s. push (temp \right); 3 return ans; 3 * Rost Orden Traversal; Post anden: 1 8 4 5 2 6 7 3 1 (LRN) be know that we have print nodes in the ander \right, RN.	
Node *temp = S. tap(): s. pap(): ans. push back (temp \right); if (temp \right) right); if (temp \right) left); s. push (temp \right); 3 return ans; 3 * Rost Orden Traversal; Post anden: 1 8 4 5 2 6 7 3 1 (LRN) be know that we have print nodes in the ander \right, RN.	
ans. push back (temp => data); if (temp => right) s. push (temp => right); if (temp => left) s. push (temp => left); 3 return ahs; 3 ** Post ander: Post ander: 1 4 1 1 1 1 1 1 1 1 1 1 1	
ans. push back (temp → data); if (temp → night) s. push (temp → night); if (temp → left) s. push (temp → left); 3 neturn ans; 3 ** Post ander: Post ander: 1 (LRN) be know that we have print pades in the ander — LRN.	
if (temp → right); s. push (temp → right); if (temp → left); s. push(temp → left); 3 return ahs; 3 ** Post Orden Traversal; Post anden: 1 1 1 1 1 1 1 1 1 1 1 1 1	
s, push (1emp > right); if (1emp -) left) s, push (1emp -) left); return abs; return abs; Post ander Traversal; Post ander: 1 8 4 5 2 6 7 3 1 (LRN) We know that we have print pades in the ander	
s. push(temp => left); 3 neturn ahs; Rost Orden Traversal; Post anden: 1 8 4 5 2 6 7 3 1 (LRN) We know that we have print pades in the ander — LRN.	
s, push(temp => left); 3 notion ans; Rost Orden Traversal; Post ander: 1 8 4 5 2 6 7 3 1 (LRN) We know that we have print pades in the ander — LRN.	
* Post Order Traversal: Post ander: \$ 4 5 2 6 7 3 1 (LRN) We know that we have print pades in the ander — LRN.	
* Post Order Traversal: Post ander:	
Post ander: B B B B C B C C C C C C C	
Post ander: B B B B C B C C C C C C C	
Post ander: 3 8 4 8 4 1 1 1 1 1 1 1 1 1 1 1 1	
3 8 4526731 (LRN) We know that we have print pades in the ander— LRN.	
8 4 5 2 6 7 3 1 (LRN) We know that we have print pades in the ander — LRN.	-
= 3 8 4 5 2 6 7 3 1 (LRN) = We know that we have print nodes in the order — LRN.	-
- (LRN) → be know that we have print nodes in the order — LRN.	-
(LRN) ⇒ We know that we have print nodes in the ander — LRN.	+
(LRN) ⇒ We know that we have print nodes in the ander — LRN.	1
the order — LRN.	1
the lander - link R. N.	
=> But I if we reverse the order - NRI	
then we can easily solve our problem.	
=> And the answer , that we will get i we	1
will reverse it that is our required	
answer.	1
- And chart is	-

Date _ Page . we will use the same provious approach. Inorder Traversal: 8 4 2 5 1 6 3 7 we have to print that element that we are visiting second time. So, when we get an node and that node is 7 first time. Then we are pushing right, node & left a local In second time, we will directly print the =) node! withhing is 10 line bolier For knowing every node visiting status, we will use an another stack. for every node, we will store their visiting status in the second stack. vector (int > inorder (Node *100+){ stack < Node * > S; stack < book > visited;

s. push (noot); visited push(0); vector (ans int) ans; while (!s.empty) { Node * temp = s.top(); s. pop(); bool flag = visited top(); if (flag == 0){ if (temp → rught) {
s. push(temp → right);
visited. push(0); s, push (temp); visited push (1); if temp > left) { s. push (temp → left) visited, push(0); 3 else { ans, push_back (temp -data); noturn ans;