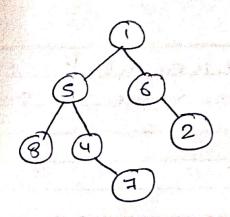
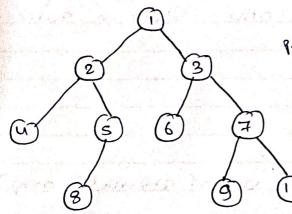
Traversal of Tree



Inorder [8,5,471,6,2] Preorder [1,5,8,4,7,6,2] PostOrder [8,7,4,5,2,6,7]

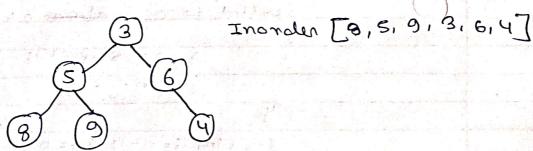


Inorder [4,2,8,5,1,6,3,9,7,10] Preorder [1,2,4,5,8,3,6,7,9,10] Post order [4,8,5,2,6,8,10,7,3,1]

Inorden :- reft -> Root -> right.
Preorden:- root -> left -> right
Post Orden:- left -> right -> root.

now, are will gonna dry run one example wing

consider the Tree :-



we will only do dry run for Inorder Traversol.

| (B. Wot, arr) 11 8 soft return. | | tracing each step of recursion. |
|--|--|--|
| Sil (root==null) Trooten Trooten (root-libt, arr); > 15, arr. arr.add (root-val) Trooten (root-right, arr); Trooten (5, arr) Carr.add (2001-val) Trooten (8, arr) Trooten (8, arr) (8.Uft, arr) / 8.tef (==null) Arr.add (8) // It will add & in arr.add (8, right, arr) Trooten (8, right, arr) S (8.vight == null) | 3 7 - | |
| Thordun (roditable, arr); > 1 5, arr. arr.add (roditable, arr); Thordun (roditable, arr); Thordun (roditable, arr); Thordun (s. arr) Thordun (s. arr) (8.44, arr) / s.b. (= arr. (8.44, arr) / s.b. (= arr. (8.44, arr) / s.b. (= arr. arr.add (8) // It will add g in Inordun (8. right, arr) own empty array Thordun (8. right, arr) (8. right = arr) | | Function Transler (root, arraylist car) |
| Thorder (roal left, err); > 1 5, arr. arriadd (roal right, arr); Trorder (roal right, arr); Trorder (5, arr) Enorder (sock right, val); Thorder (8, arr) [8.44, arr) // 8.44 (= = nath) Thorder (8, arr) [8.44, arr) // 8.44 (= = nath) 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | | il (rost = = null) |
| Inorder (roditely, arr); S. arr. arriadd (rodinght, arr); Inorder (rodinght, arr); Inorder (s. arr) Enorder (sock right, val); Inorder (sock right, val) Inorder (sock right, arr) | | |
| Thordus (real right, arr); Thordus (s. arr) (8, arr) (8, arr) Thordus (sood right, val); Thordus (sood right, arr) Thordus (sood right, arr) (8. Uft, arr) Thordus (sood right, arr) (8. Uft, arr) Thordus (sood right, arr) (9. Vight = 2 null) | | return |
| Trorder (rod right, arr) Trorder (s. arr) (8, arr) (8, arr) Trorder (soct right, val); (8. Uft, arr) (8. Uft, arr) Trorder (8, arr) 3 Trorder (8, right, arr) Trorder (8, right, arr) Trorder (8, right, arr) Trorder (8, right, arr) | | Inordu (rodilett, arr):>11 5, arr. |
| Incredu (5, ass) (8, ars) (9, ars) (10, ars) (11, ars) (12, ars) (13, ars) (14, ars) (15, ars) (15, ars) (15, ars) (16, ars) (17, ars) (18, ars) (19, ars) (10, ars) (11, ars) (11, ars) (12, ars) (13, ars) (14, ars) (15, ars) (15, ars) (16, ars) (17, ars) (17, ars) (18, ars) (18, ars) (19, ars) (10, a | S | arriadd (rootival) |
| Incredu (5, ass) (8, ars) (9, ars) (1, ars) (1, ars) (1, ars) (2, ars) (3, ars) (4, ars) (5, ars) (6, ars) (7, ars) (8, ars) (8, ars) (9, ars) (9, ars) (1, ars) (1, ars) (1, ars) (2, ars) (3, ars) (4, ars) (5, ars) (6, ars) (7, ars) (8, ars) (8, ars) (9, ars) (9, ars) (1, ars) (1, ars) (1, ars) (2, ars) (3, ars) (4, ars) (5, ars) (6, ars) (7, ars) (8, ars) (8, ars) (8, ars) (9, ars) (9, ars) (1, ars) (1, ars) (1, ars) (1, ars) (2, ars) (3, ars) (4, ars) (4, ars) (5, ars) (6, ars) (7, ars) (8, ars) (8, ars) (9, ars) | | Inordin (rodiniant, and); |
| (8, arr) (8, arr) (8, arr) (8, arr) (100000000) Inorder (8, arr) (8. Uft, arr) (8. Uft, arr) (8. Vight, arr) (8. Vight, arr) Inorder (8. right, arr) (8. right, arr) (8. right = 2 n U11) | | Q |
| (8, arr) (8, arr) (8, arr) (8, arr) (100000000) (100000000) (1000000000) (10000000000 | K D | 3 |
| (8, arr) (8, arr) (8, arr) (8, arr) (100000000000000000000000000000000000 | 1 | 4 |
| (8, ars) Crr. add (2001-001); Inordur (8, ars) [8.44, ars) // 8.46 (== nather) [8.44, ars) // 30, return arr. add (8) // It will add & in 3 Inordur (8. right, arr) own empty array Inordur (8. right, arr) [8.44, arr) [9.44, arr) [| | Inordu (Siarr) |
| Inordur. (**ook. right, val); Inordur. (**ook. right, val); Inordur. (**anr) (**B. Uft, anr) // **. left = = nath | ja . | S |
| Thorder (8, ans) [B.Uft, ans) // 8. left = = nate [B.Uft, ans] // 30, return [arr.add(8) // It will add 8 in [arr.add(8) // | | (gars) > Hoto mall 30 |
| Inordur (*ost. right, val); Inordur (8, anr) (8. Uft, arr) // 8. lef (== nate of so, return) arr. add (8) // It will add g in Inordur (8. right, arr) our empty array Inordur (8. right, arr) (8. vight == null) | gal. | crriadd (swalves) |
| Inorder (8, ans) [8.44, ans) // 8.44 == nate [8.44, ans) // 8.44 == nate 3 arr.add (8) // It will add 8 in Inorder (8. right, arr) our empty array Inorder (8. right, arr) [8. vight == null] | | Inordu. (root. right, val); |
| (8.Uft, arr) 11 8.205 return arr.add(8) 11 It will add g in Inordun(8.right, arr) own empty array Inordun(8.right, arr) S (8.right == hull) | | - O I HOLD IN THE STATE OF THE |
| (8. Wft, arr) 1/8. Set return arr.add (8) 1/ It will add & in Inorder (8. right, arr) our empty array Inorder (8. right, arr) S (8. right = = null) | And the state of t | |
| (8.Uft, arr) 11 8.205 return arr.add(8) 11 It will add g in Inordun(8.right, arr) own empty array Inordun(8.right, arr) S (8.right == hull) | | Inorda (8, anr) |
| (8.Uft, arr) 1/8.20th return arr.add(8) 1/ It will add & in Inordun(8.right, arr) own empty array Inordun(8.right, arr) S (8.right == hull) | | The state of the s |
| arr.add(8) // It will add & in I Inordun (B. night, arr) own empty array Inordun (B. right, arr) S (B. right == null) | | 2 10 116t, arr / / 8. left = = nall |
| Inorder (8. right, arr) our empty array Inorder (8. right, arr) S (8. right == null) | * | |
| Inorder (8. right, arr) S (8. right = = null) | | 020-04-02001110 |
| 5 (8. right = = null) | | 3 Inoraul (Binightiari) |
| 5 (8. right = = null) | | |
| S (8. right = = null) S returner | X25 | |
| 2 retioner | | S (8. right = = null) |
| | | e returns |
| 7 | | 74 4 |

```
Inorder (Scarr)
           In Order (8, ar):
                               It Is done
          arr. add (root.val)
                               list has one
         11 It will add 5 also.
                               element 8
        Inorder (5. Lebs, arr
       arr= [8,5]
  Inorder (s. left, arr)
11 be course the grecursion will go one step &
                     return back with
        arr. add (g);
  Il Recursion will go right, but it is now.
   arr = [8,5,9,
   Inorder (3, anr)
        Inorder(3-left, arr): // It is don.
       arroad (3): [815,913]
Inorder (3. right, arr).
   Inorder (6, arr)
       [6. left == neel ) so, it will return.
        arr. add (6); [8,5,9,3,6].
          Inordu (6. right, am)
```

Troader (4, arr) THE right is also new. Moces, this function will removed from stack, Each bunetion will seemone Step by Step. Inorder (3,000 Inorder(3-left, arr) // done arr. add (3); Ildono. Inorder (3. right, arr) // done so, it will be turn to main fun where our has all the nocles [8,5,9,3,6,4]