



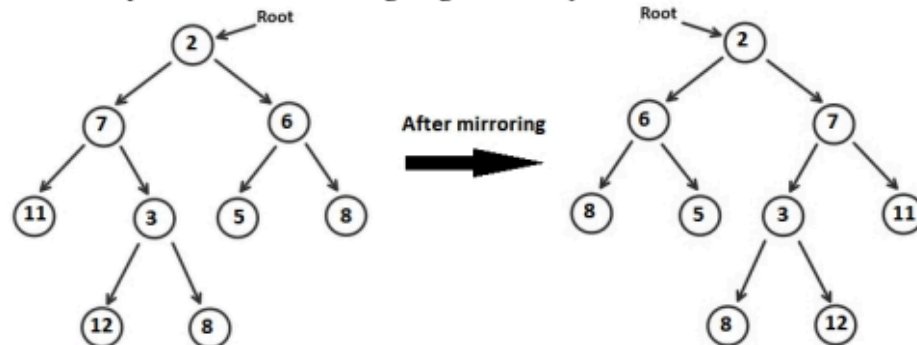
United International University
Dept. of Computer Science & Engineering
Course Title: Data Structure and Algorithm-I Laboratory
Section: A
Course Teacher: Siana Rizwan

Trimester 203

- c) Consider the **treenode** structure given below to create a binary tree. [1.5]

```
struct treenode {  
    int data ;  
    struct treenode *left ;  
    struct treenode *right ;  
};
```

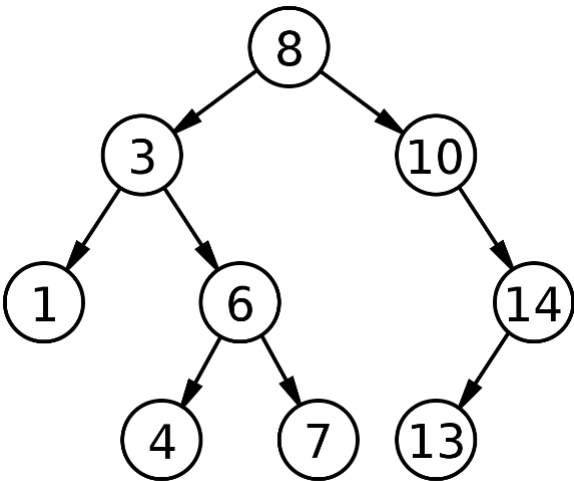
Write a recursive C function `void mirror_tree(struct treenode* t)` that will mirror the entire binary tree. See the following diagram for explanation.



Hints: Swap the pointers using postorder traversal technique.

1. Print Nodes at a Specific Depth of BST

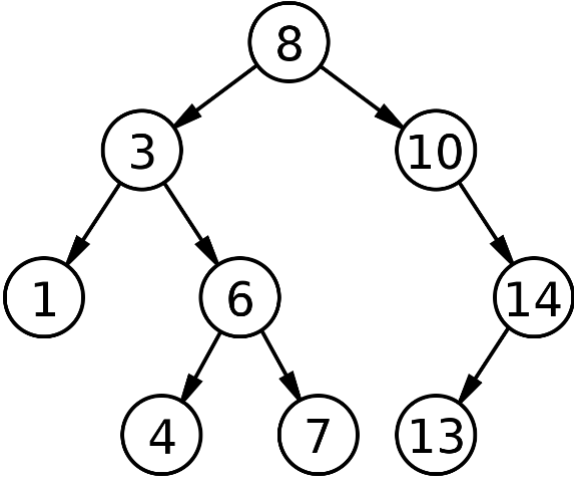
You work in HR and have the company's organizational chart stored as a BST based on employee IDs. You need to find all employees at a specific management level.

Sample Input	Sample Output
<div><pre>graph TD; 8((8)) --> 3((3)); 8 --> 10((10)); 3 --> 1((1)); 3 --> 6((6)); 6 --> 4((4)); 6 --> 7((7)); 10 --> 14((14)); 14 --> 13((13));</pre></div> <div>No of id: 9 Values of each id: 8 3 10 1 6 14 4 7 13 Depth = 2</div>	<div>Employees at depth 2: 1, 6, 14</div>
<div>No of id: 9 Values of each id: 8 3 10 1 6 14 4 7 13 Depth = 3</div>	<div>Employees at depth 3: 4, 7, 13</div>

[Depth of root node is 0 and depth increase by going downwards of tree]

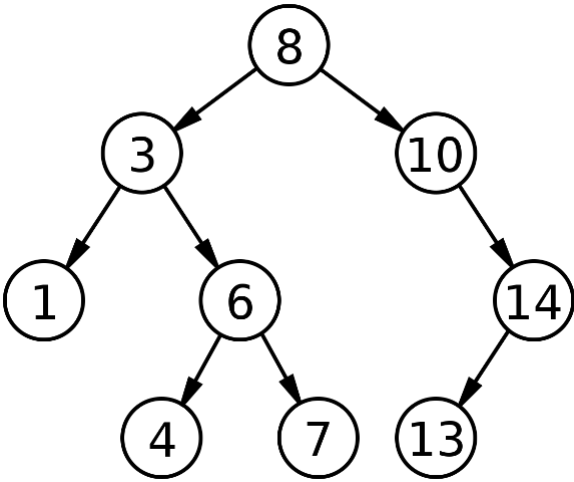
2. Find the maximum value from a BST

Your company maintains employee salaries in a BST. Find the employee with the highest salary.

Sample Input	Sample Output
 <pre>graph TD; 8((8)) --> 3((3)); 8 --> 10((10)); 3 --> 1((1)); 3 --> 6((6)); 6 --> 4((4)); 6 --> 7((7)); 10 --> 14((14)); 14 --> 13((13));</pre> <p>No of employee: 9 Salary of each employee: 8 3 10 1 6 14 4 7 13</p>	Highest salary: 14
<p>No of employee: 9 Salary of each employee: 8 11 12 20 6 12 4 7 17</p>	Highest salary: 20

3. Calculate height of a BST

Your company wants to know how many management levels exist from CEO to entry-level employees.

Sample Input	Sample Output
 <pre>graph TD; 8((8)) --> 3((3)); 8 --> 10((10)); 3 --> 1((1)); 3 --> 6((6)); 6 --> 4((4)); 6 --> 7((7)); 10 --> 14((14)); 14 --> 13((13));</pre> <p>No of nodes: 9 Values of each node: 8 3 10 1 6 14 4 7 13</p>	<p>Levels : 4</p>

Best of Luck!