Assignment - 4

Name	:	B. Ravi Kumar
Reg NO	:	192311246
Course	:	Data structure for stack overflow.
Course code	:	CSA0389.

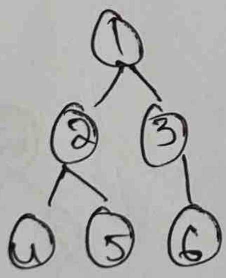
1) Develop a C Program to implement the Tree traversals (Inorder, Preorder, Post order).

```
#include <stdio.h>
#include <stdlib.h>.
Structure node {
        int data ;
        struct node * Left ;
        struct node * Right ;
};
struct node * create node (int data) {
struct node * node = (struct node*) malloc (size of
                    (struct node));
  new node → data = data ;
  New node → Left = Null ;
  new node → right = Null ;
  return newnode ;
}
void traversal (struct node * root) {
   if (root = = Null).
      return ;
   in order Traversal (root → Left);
```

```
    Print f (" %d ", root → data);
    inordertraversal (root → right);
 }
 void Preorder Traversal (struct node* root ) {
      if (root == NULL)
        return ;
      Print f (" %d " root → data);
     Preorder traversal (root → Left);
     Preorder traversal (root → right);
 }
  void Postorder traversal (struct node *root ) {
      if (root == NULL)
        return ;
      Postorder traversal (root → Left);
      Post order traversal (root → Right);
       Print (" %d " , root → data);
 }
  int main () {
    struct node * root = Create node (1);
     root → Left = Create node(2);
     root → Right = Create node (3);
     root → Left → left = Create node (4);
     root → Left → right = Create node (5);
```

```
Printf (" Inorder traversal : ");
inorder traversal (root);
 Printf ("\n");
Printf (" Preorder Traversal : ");
 Preorder Traversal (root);
  Printf ("\n");
 Printf(" Postorder Traversal : ");
Postorder Traversal (root);
  Printf ("\n");
   return 0;
}
```

Input: create the tree.



Output:

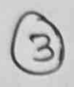
Inorder : 4, 2, 5, 1, 3, 6

Preorder : 1, 2, 4, 5 1 3, 6

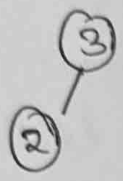
Postorder : 4, 5, 2, 6, 3, 1.

② construct An AVL Tree for 3,2,1,4,5,6,7,10,16.

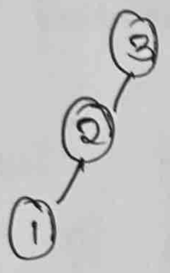
Insert 3



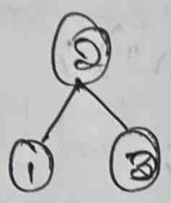
Insert 2.



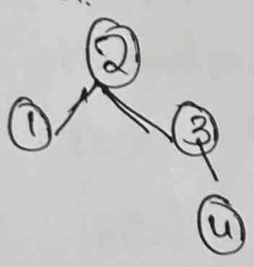
Insert 1



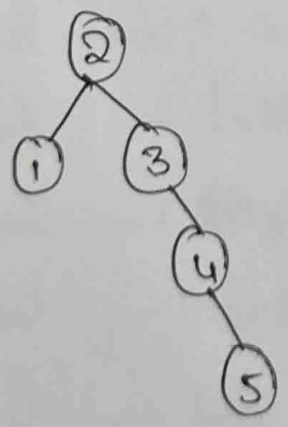
$\longrightarrow$



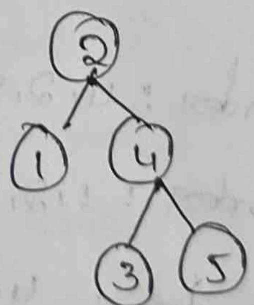
Insert 4.



Insert 5

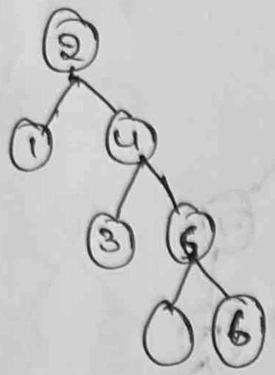


$\longrightarrow$

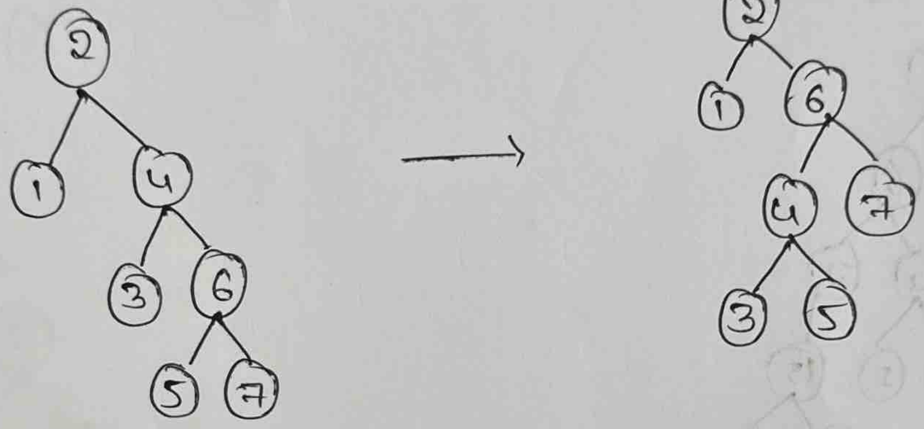


16.

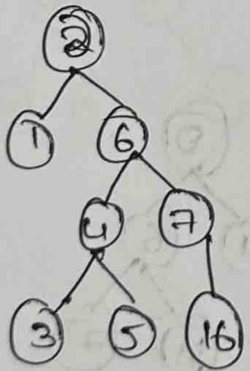
Insert 6



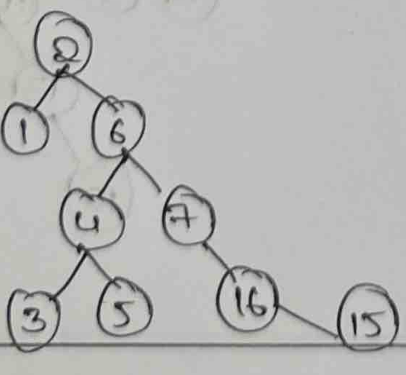
Insert 7.



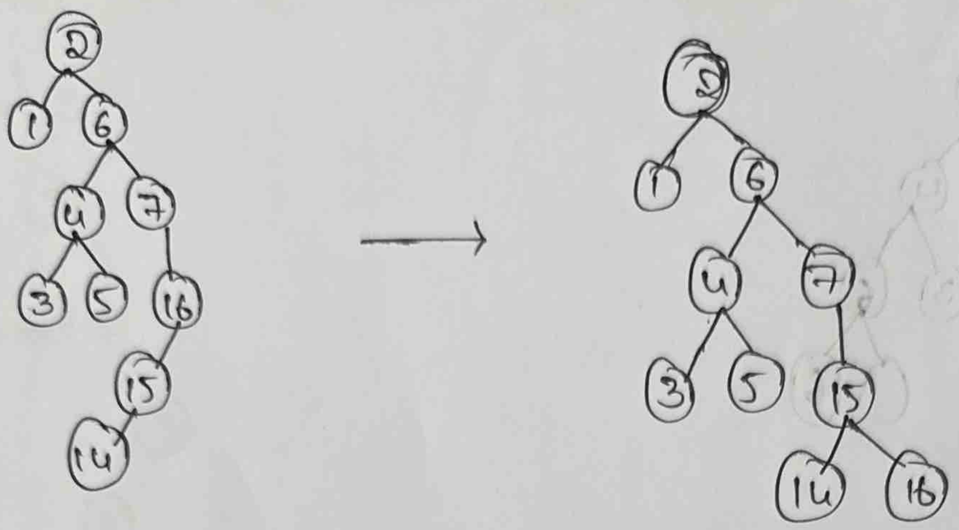
Insert 16



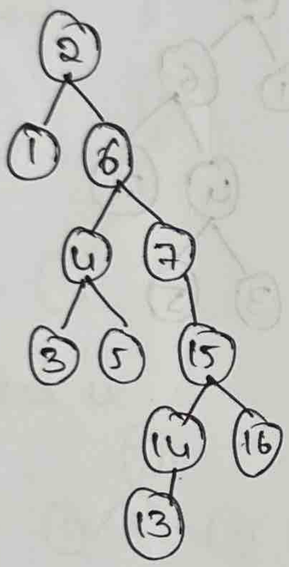
Insert 15



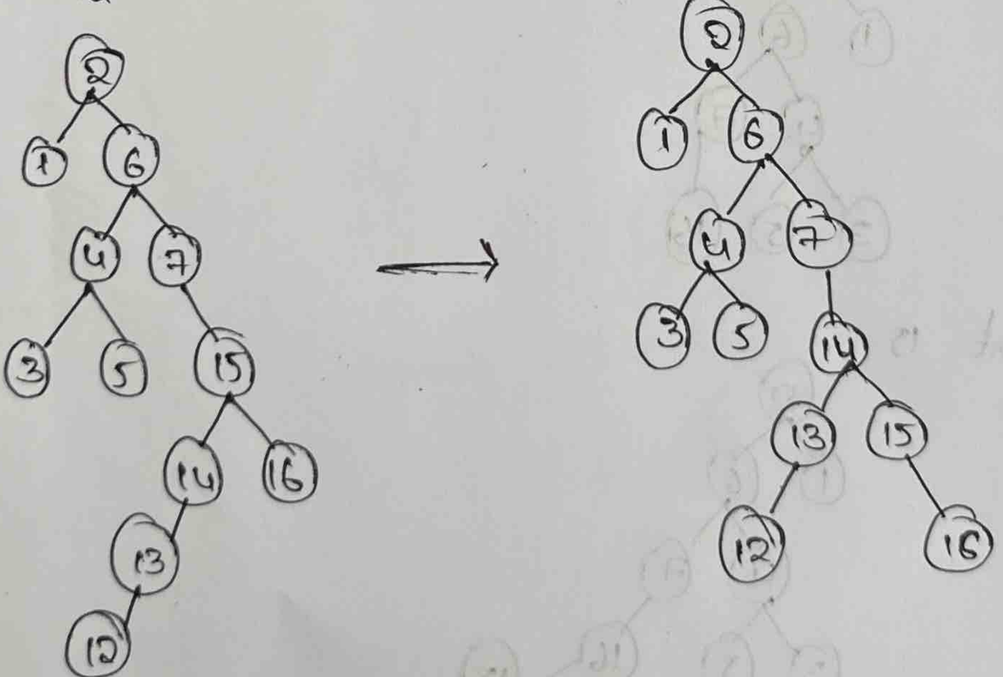
Insert 14



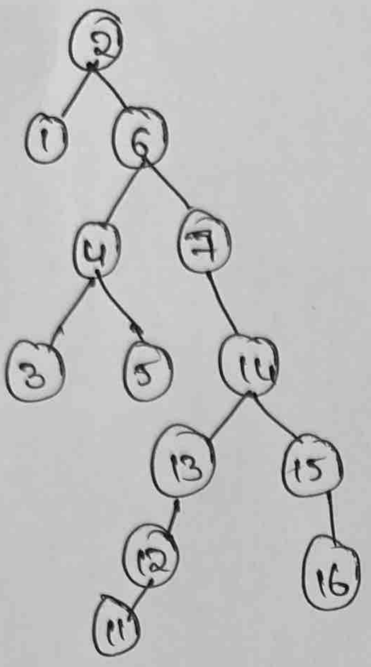
Insert 13



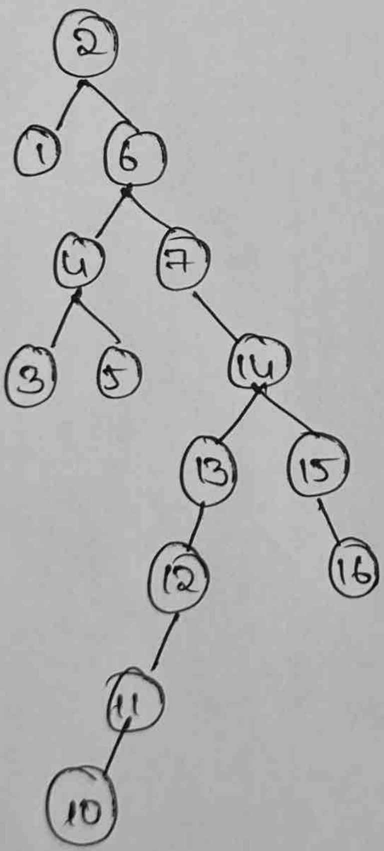
Insert 12



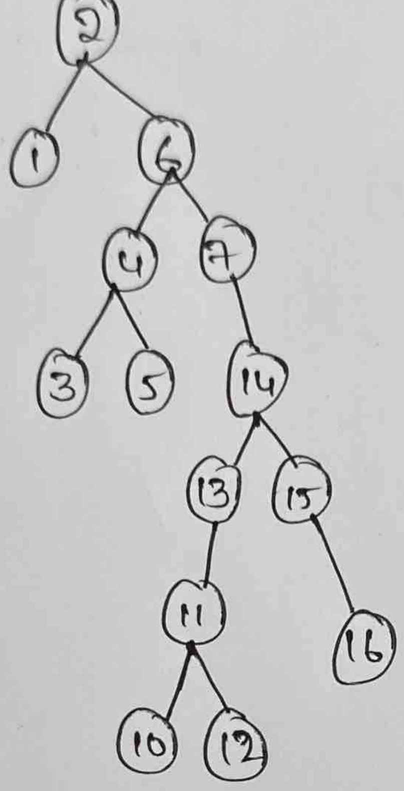
Insert 11.



Insert 10



After rotation Final Tree :



∴ Balanced.