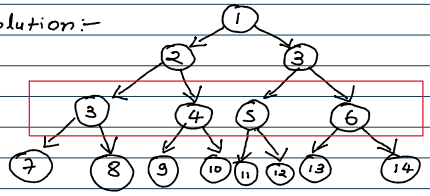
Now we have another question :-

Q) Problem Statement: Traverse the tree and Print All the elements of Given Level?

Solution :-



Suppose we have to Print All the elements under Red Border...

Now we could solve the problem by pre, in, or post method...

My thought : (I would Solve the problem By Infix APPROACH)

I would Call left Subtree BY using Recursive Approach (See figure Given Below)

SABSE Pehle hum root ko call karege Jo ki Abhi Currently (1) hi hai...

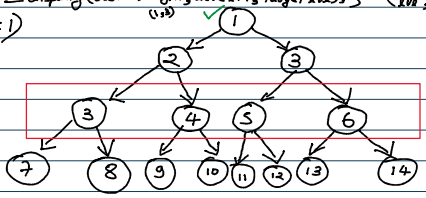
Function's Structure :-

```
Void level_display (TreeNode* root, int lvl, int target level){
  if(root == Null) return;
  if( level == Target_level){
      Cout<< root->Val;
      return; //for Optimisation}
  level_display(root->left, level+1, target lvl);
  level_display(root->right, level+1, target lvl); }
```

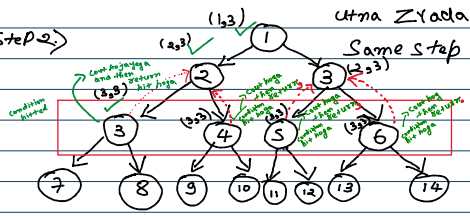
(lvl, target lvl)

Hum log ye saare kaam ek function ke through karege Yani ek recursive approach lene ke liye ek function Banayege.
nth_level_display(TreeNode* root, int lvl, int target_lvl);

Step : 1)



Step 2:



Utna Zyada Step nahi likhege wahi Sab Same step hoga...

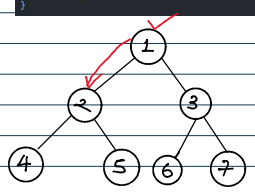
Now the function has completed its work...

Now I would Put over a snapshot of my code and its complete Output...

Assuming target :=2 (0 Based Level Indexing)

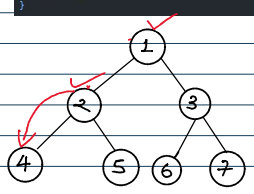
Step: 1)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



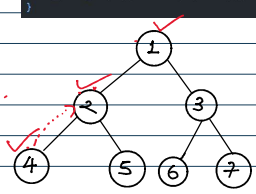
Step: 2)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



Step: 3)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



Output Screen

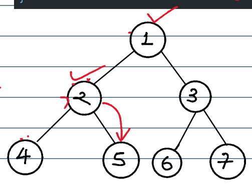
4 5 6 7

Step: 4)

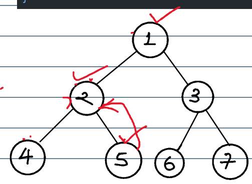
Step: 5)

Step: 6)

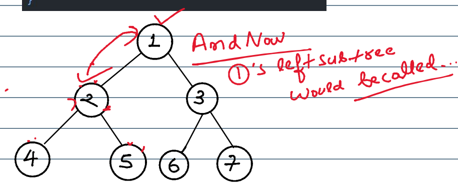
```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```

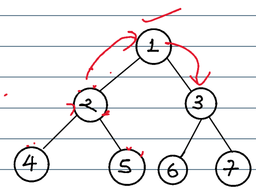


```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



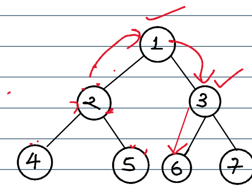
Step:3)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



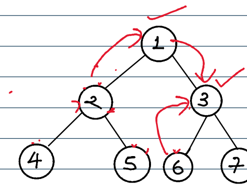
Step:3)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



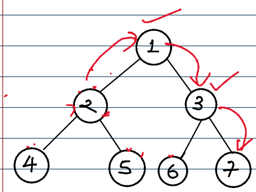
Step:3)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



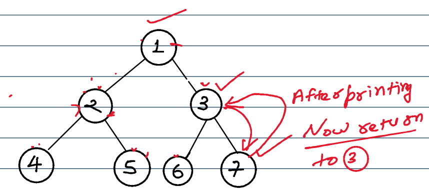
Step:4)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



Step:4)

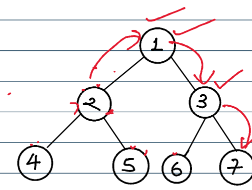
```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```



Step:4)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```

Return Back to ①.



Step:6)

```
void nth_level_display(node* root,int lvl,int t_lvl){
  if(root == NULL) return;
  if(lvl == t_lvl){
    cout<<root->val<<" ";
    return;
  }
  nth_level_display(root->left,lvl+1,t_lvl);
  nth_level_display(root->right,lvl+1,t_lvl);
}
```