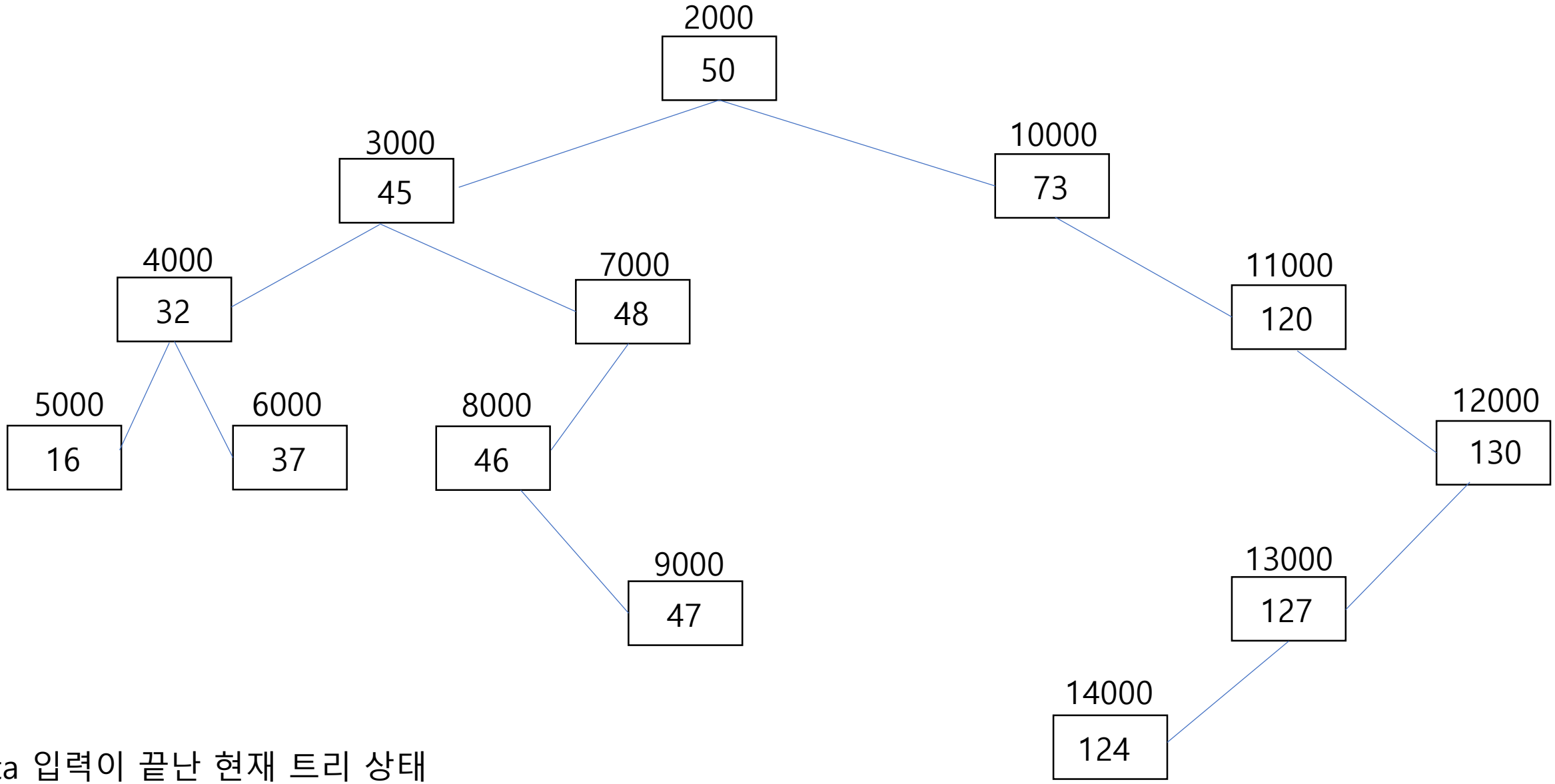
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 문한나
mhn97@naver.com

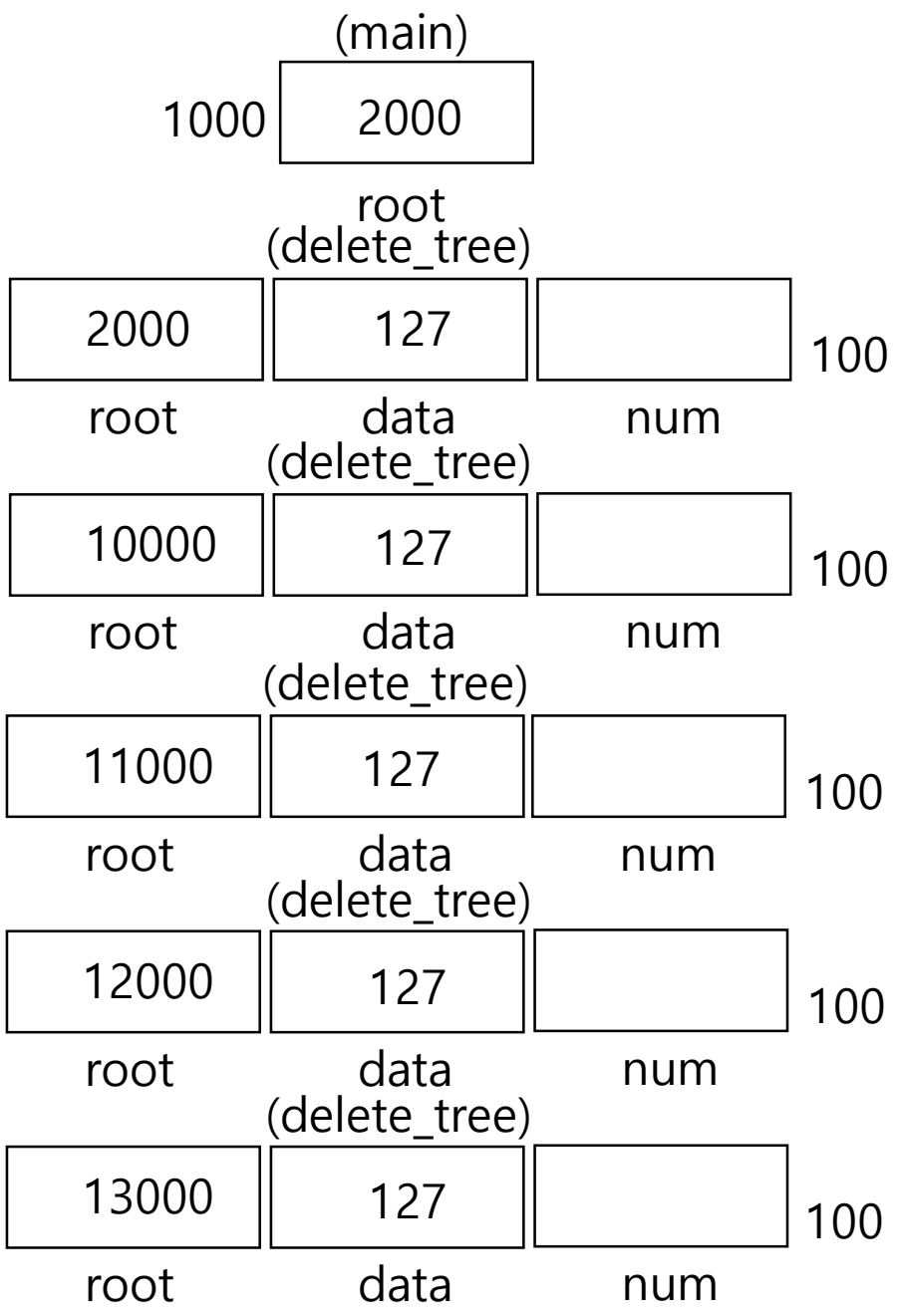
연결리스트 예제 그림 그리기 – tree 복습



data 입력이 끝난 현재 트리 상태

1. Delete 127

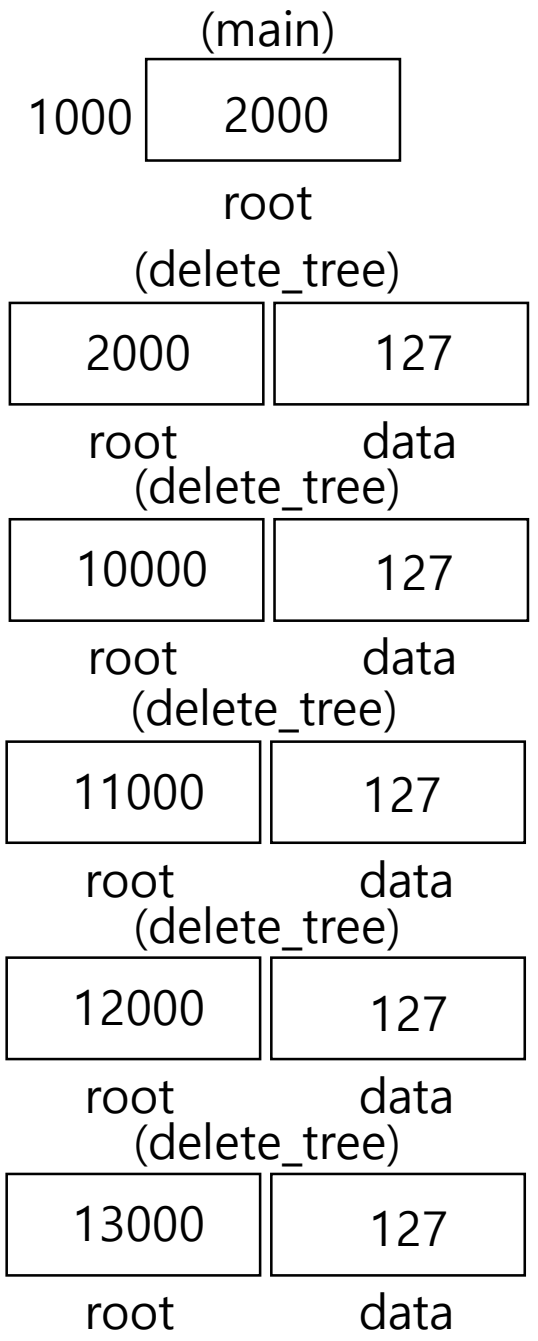
```
tree *chg_node(tree *root){
        tree *tmp = root;

        if(!root->right)
                root=root->left;
        else if(!root->left)
                root=root->right;

        free(tmp);

        return root;

}

tree *find_max(tree *root,int *data){
        if(root->right)
                root->right=find_max(root->right,data);
        else {
                *data = root->data;
                root = chg_node(root);
        }
        return root;
}

tree *delete_tree(tree *root,int data){

        int num;
        tree *tmp;
        if(root == NULL){
                printf("Not found\n");
                return NULL;
        }
        else if(root->data>data)
                root->left = delete_tree(root->left,data);
        else if(root->data<data)
                root->right = delete_tree(root->right,data);

        else if(root->left && root->right)
        {
                root->left=find_max(root->left,&num);
                root->data = num;
        }
```

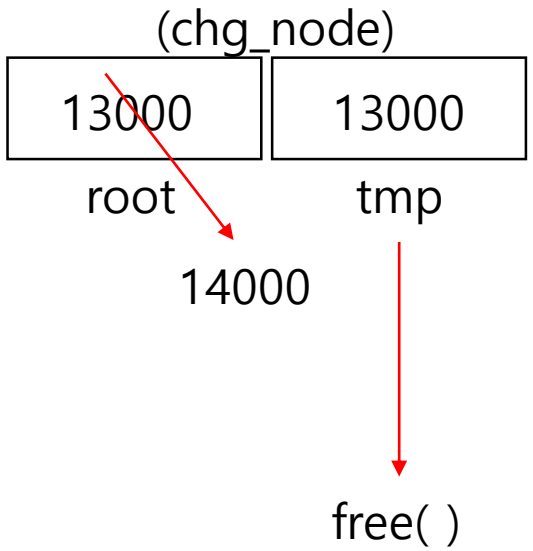


1. Delete 127

```
tree *chg_node(tree *root){
        tree *tmp = root;

        if(!root->right)
                root=root->left;
        else if(!root->left)
                root=root->right;

        free(tmp);

        return root;
```

```
tree *delete_tree(tree *root,int data){

        int num;
        tree *tmp;
        if(root == NULL){
                printf("Not found\n");
                return NULL;
        }
        else if(root->data>data)
                root->left = delete_tree(root->left,data);
        else if(root->data<data)
                root->right = delete_tree(root->right,data);

        else if(root->left && root->right)
        {
                root->left=find_max(root->left,&num);
                root->data = num;
        }
        else
                root = chg_node(root);
        return root;
}
```

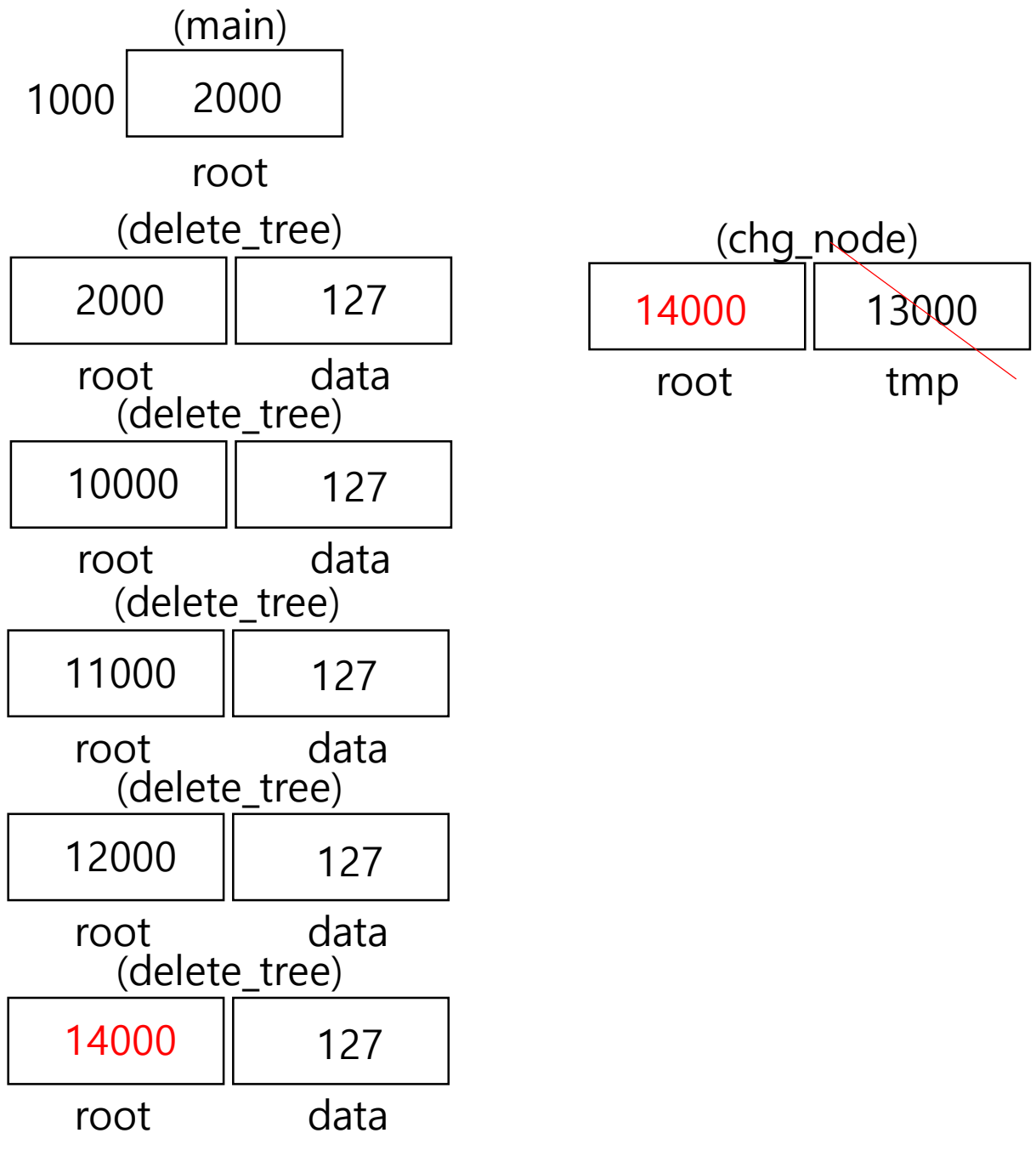


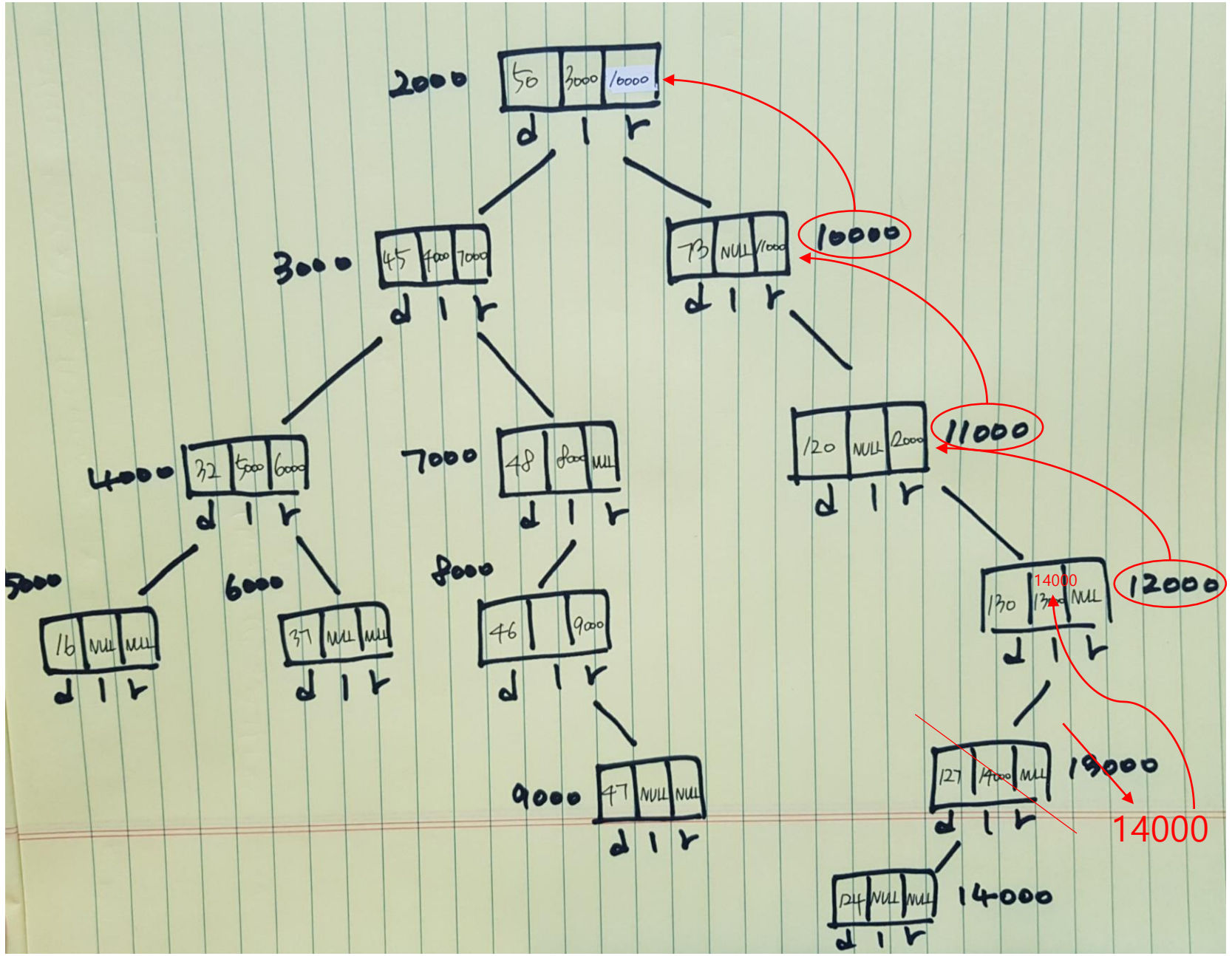


1. Delete 127

```
tree *chg_node(tree *root){
        tree *tmp = root;

        if(!root->right)
                root=root->left;
        else if(!root->left)
                root=root->right;

        free(tmp);

        return root;
```

```
tree *delete_tree(tree *root,int data){

        int num;
        tree *tmp;
        if(root == NULL){
                printf("Not found\n");
                return NULL;
        }
        else if(root->data>data)
                root->left = delete_tree(root->left,data);
        else if(root->data<data)
                root->right = delete_tree(root->right,data);

        else if(root->left && root->right)
        {
                root->left=find_max(root->left,&num);
                root->data = num;
        }
        else
                root = chg_node(root);
        return root;
}
```



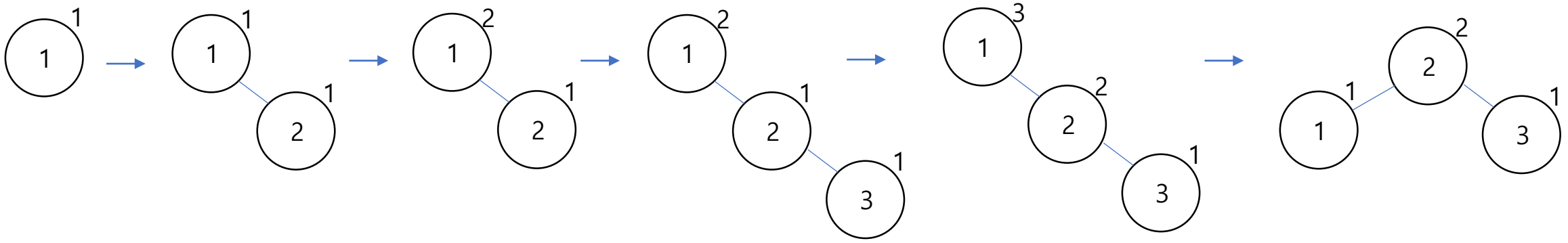



```
.mhn@mhn-900X3L:~/my_proj/c/10_s$ ./a.out
data = 50, left = 45, right = 73
data = 45, left = 32, right = 48
data = 32, left = 16, right = 37
data = 16, left = NULL, right = NULL
data = 37, left = NULL, right = NULL
data = 48, left = 46, right = NULL
data = 46, left = NULL, right = 47
data = 47, left = NULL, right = NULL
data = 73, left = NULL, right = 120
data = 120, left = NULL, right = 130
data = 130, left = 127, right = NULL
data = 127, left = 124, right = NULL
data = 124, left = NULL, right = NULL
after delete
data = 50, left = 45, right = 73
data = 45, left = 32, right = 48
data = 32, left = 16, right = 37
data = 16, left = NULL, right = NULL
data = 37, left = NULL, right = NULL
data = 48, left = 46, right = NULL
data = 46, left = NULL, right = 47
data = 47, left = NULL, right = NULL
data = 73, left = NULL, right = 120
data = 120, left = NULL, right = 130
data = 130, left = 124, right = NULL
data = 124, left = NULL, right = NULL
mhn@mhn-900X3L:~/my_proj/c/10_s$
```

AVL Tree

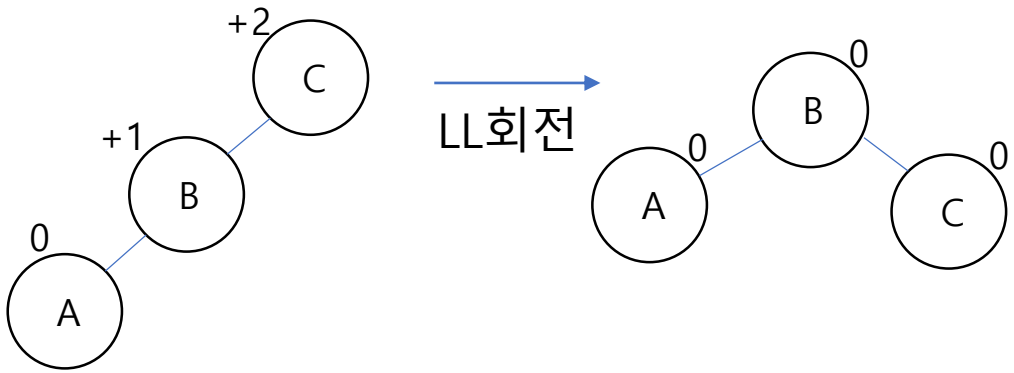
AVL트리 : 한 노드를 중심으로 좌우 종속 트리의 높이 차가 1 이하인 균형 잡힌 트리
AVL트리는 이진 트리의 삽입, 삭제를 계속할 때 어느 한 방향으로 치우치거나, 높이 차이로 인해서
수행시간이 증가되는 것을 막기 위해 균형을 유지하도록 한 것이다

만약 삽입, 삭제 후 2이상의 균형 인수차이가 나게 된다면 삽입,삭제 된 노드에서 가장 가까운 조상 노드의
서브 트리들에 대해 다시 균형을 맞추기 위한 재배치를 실시한다.



AVL트리가 깨지는 총 4가지 경우(N : 새로 삽입된 노드, A : N의 조상으로 균형인자가 ±2 되는 노드)

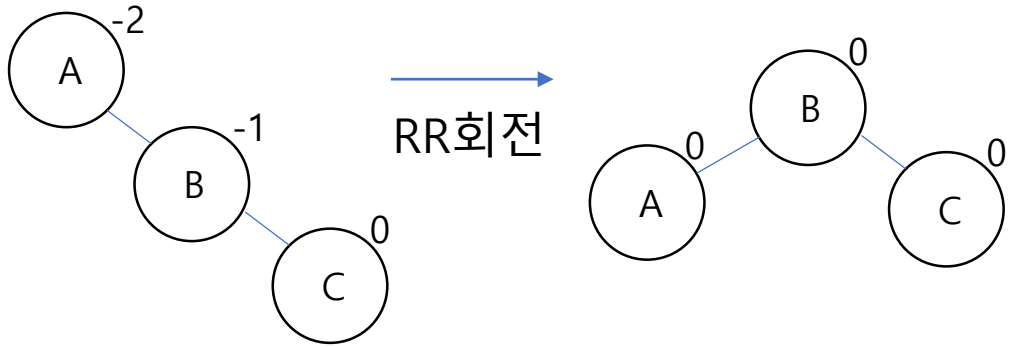
LL : N 가 A의 왼쪽의 왼쪽 부속 트리에 삽입됐을 때



오른쪽 회전을 시키면 다시 균형 회복

AVL트리가 깨지는 총 4가지 경우(N : 새로 삽입된 노드, A : N의 조상으로 균형인자가 ±2 되는 노드)

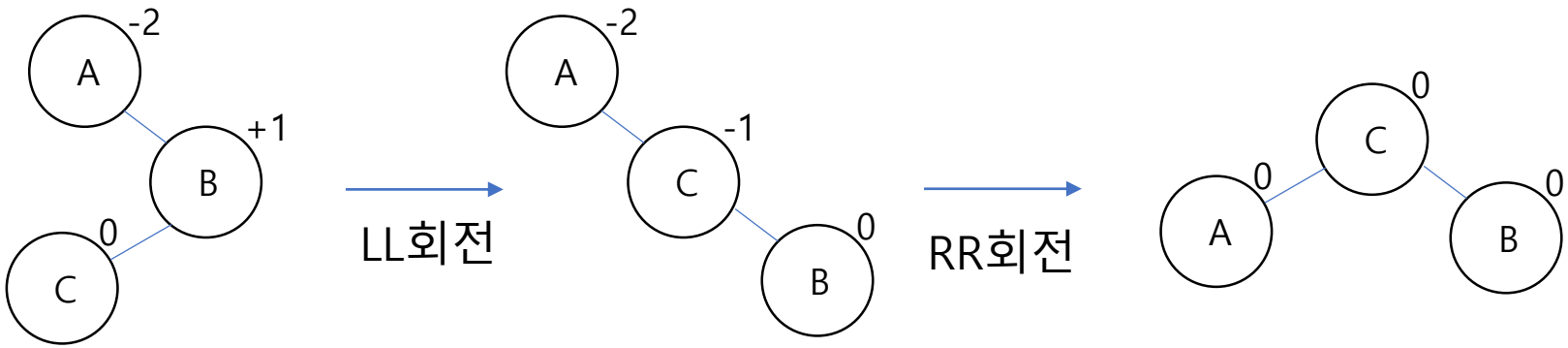
RR : N 가 A의 오른쪽의 오른쪽 부속 트리에 삽입됐을 때



왼쪽 회전을 시키면 다시 균형 회복

AVL트리가 깨지는 총 4가지 경우(N : 새로 삽입된 노드, A : N의 조상으로 균형인자가 ±2 되는 노드)

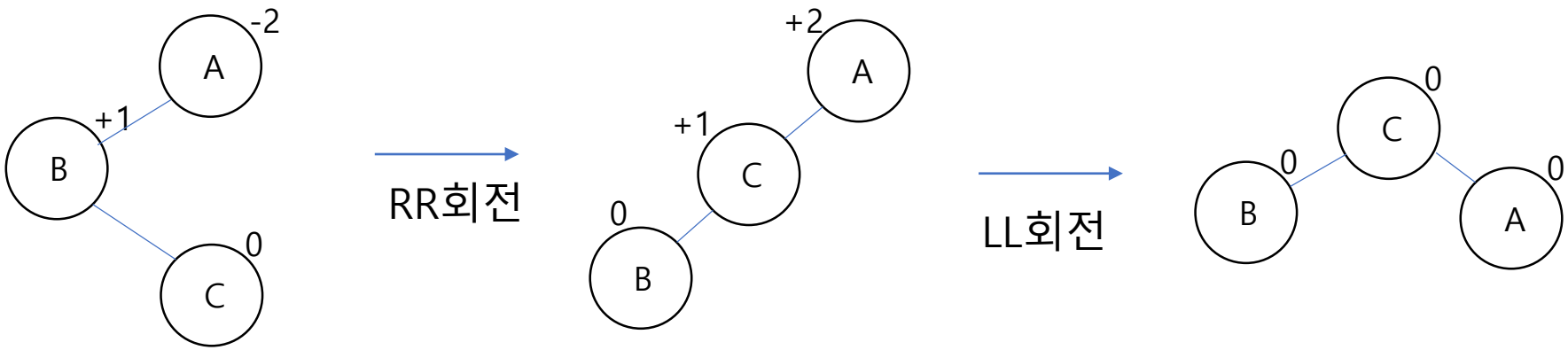
RL : N 가 A의 오른쪽의 왼쪽 부속 트리에 삽입됐을 때



오른쪽 회전 후 왼쪽 회전을 시키면 다시 균형 회복

AVL트리가 깨지는 총 4가지 경우(N : 새로 삽입된 노드, A : N의 조상으로 균형인자가 ±2 되는 노드)

LR : N 가 A의 왼쪽의 오른쪽 부속 트리에 삽입됐을 때



왼쪽 회전 후 오른쪽 회전을 시키면 다시 균형 회복