TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

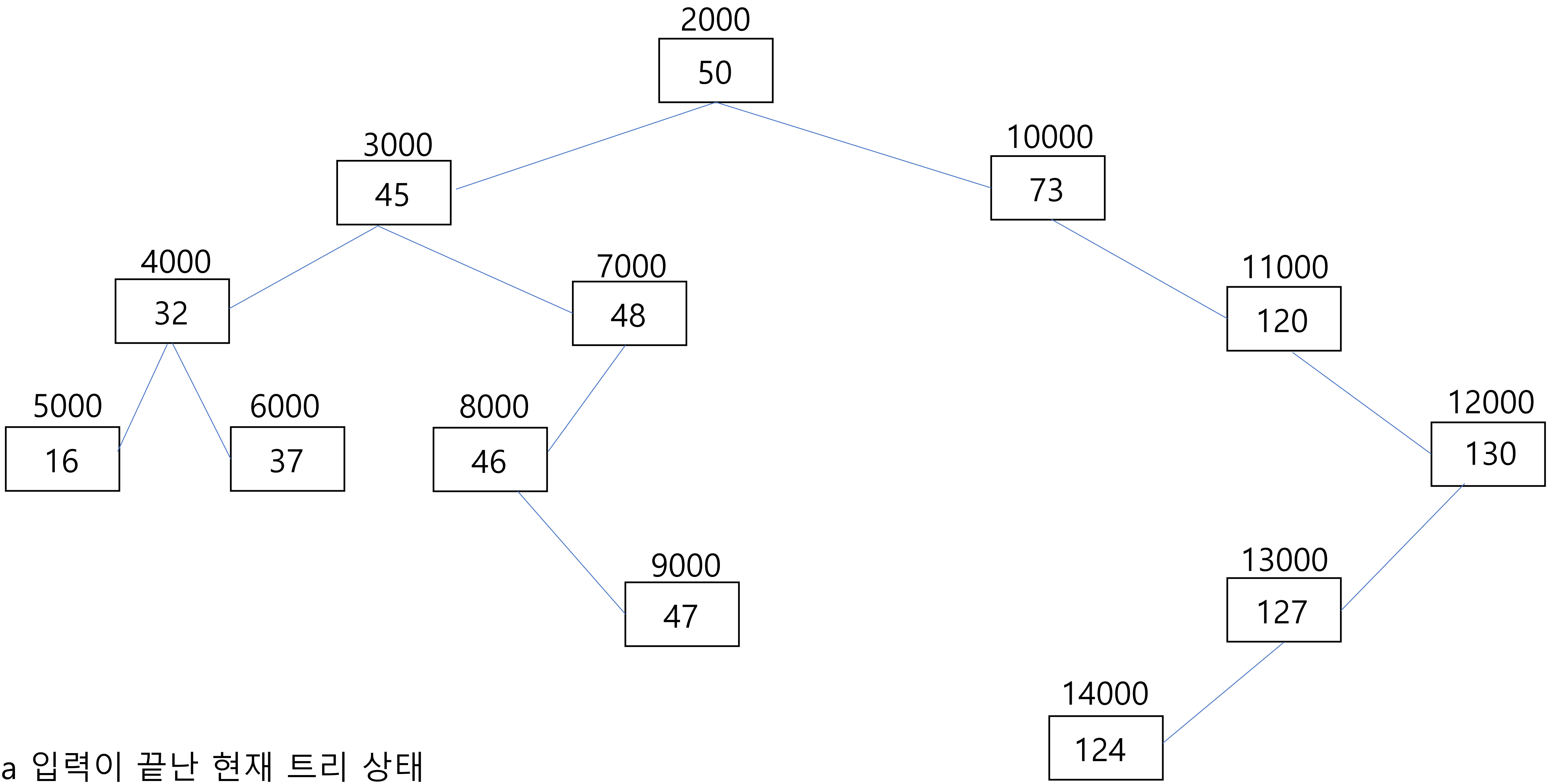
강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 문한나

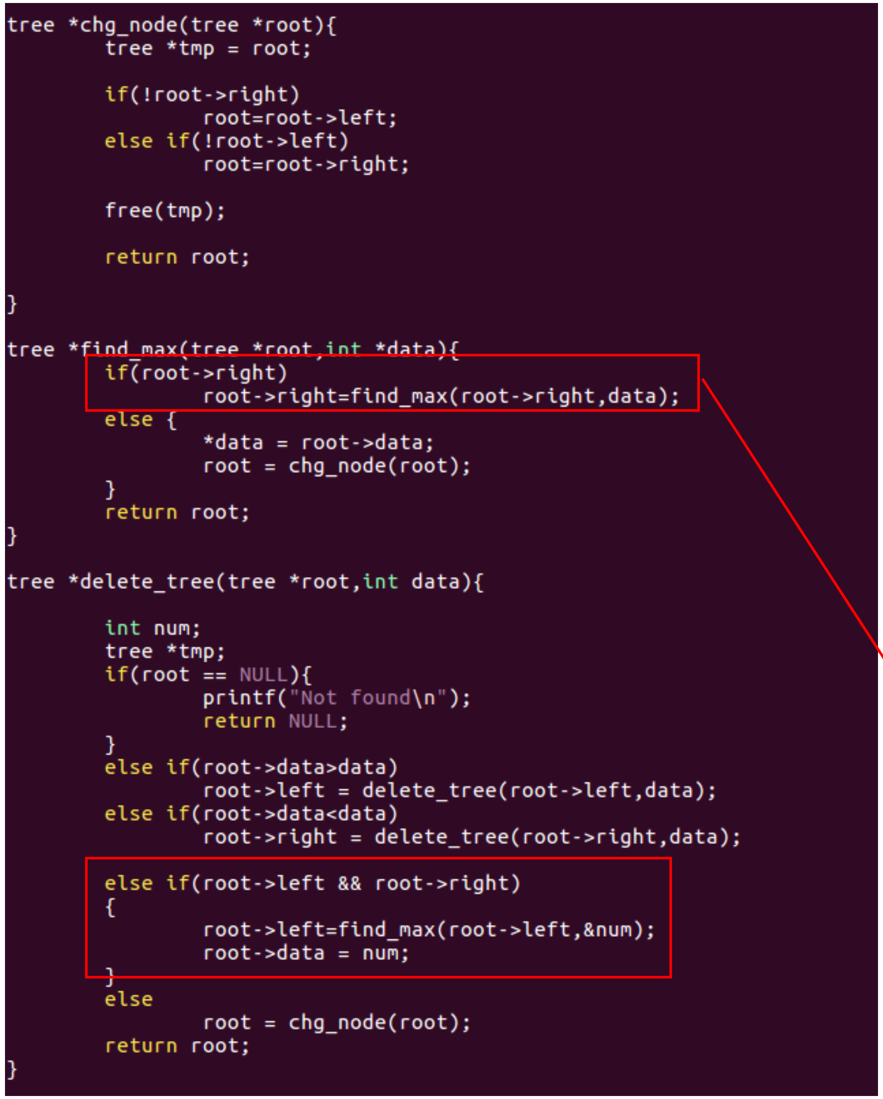
mhn97@naver.com

연결리스트 예제 그림 그리기 – tree



data 입력이 끝난 현재 트리 상태

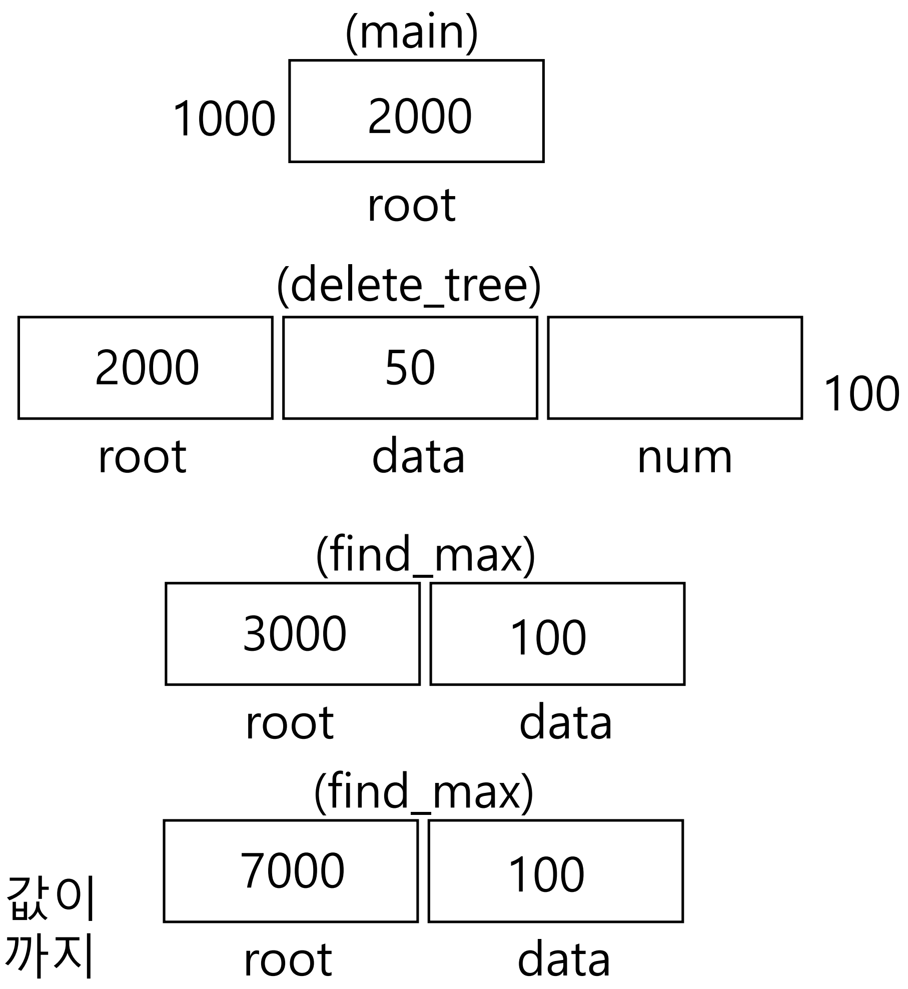
1. Delete 50

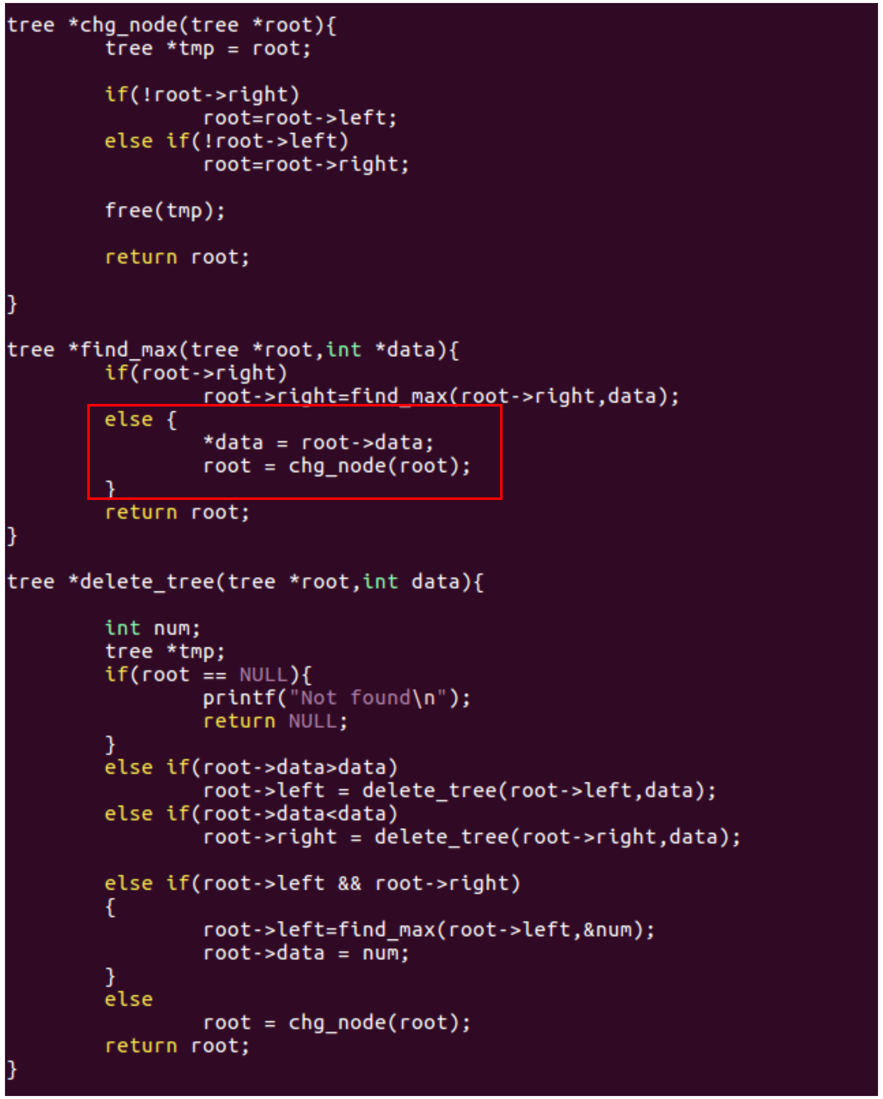


```
tree *chg_node(tree *root){
        tree *tmp = root;

        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;
        }
        else
                root = chg_node(root);
        return root;
}
```

Right의 값이
없을 때 까지
계속 호출한다

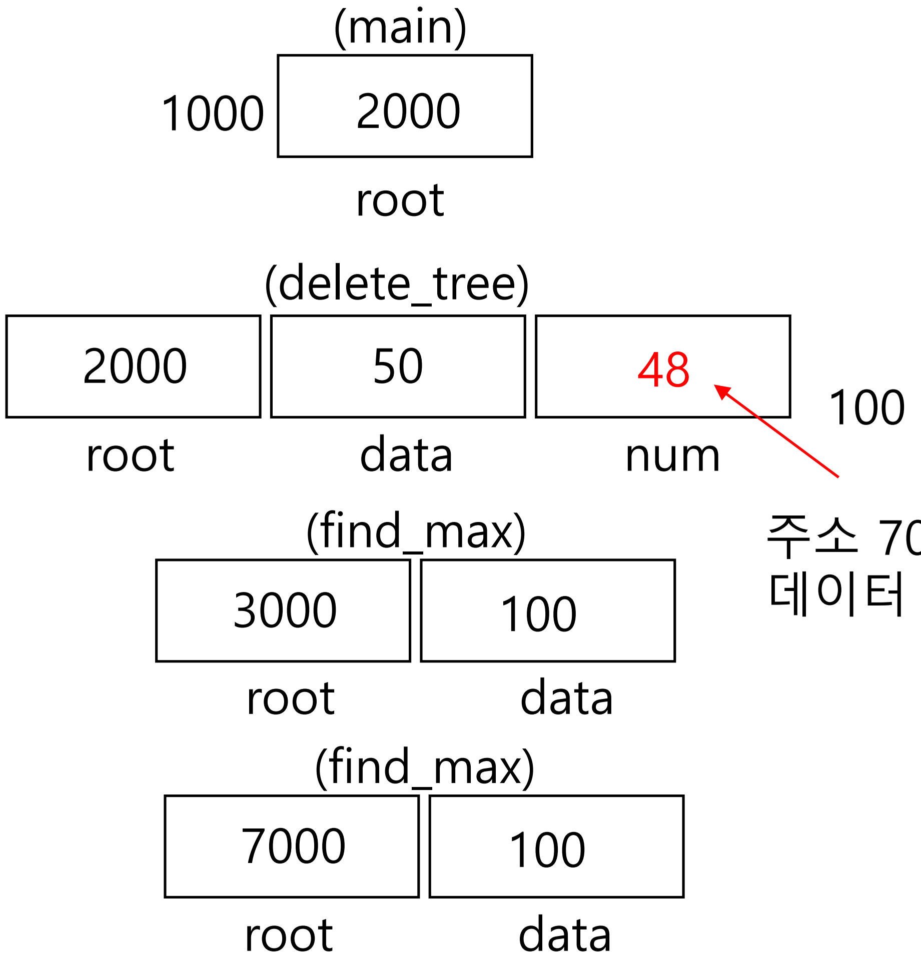
left와 right값
둘 다 있을 때
find_max호출





```
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;
        }
        else
                root = chg_node(root);
        return root;
}
```

주소 7000에
right값이
없으니
else 실행



(main)

1000 | 2000 (root)

(delete_tree)

2000 (root) | 50 (data) | 48 (num) 100

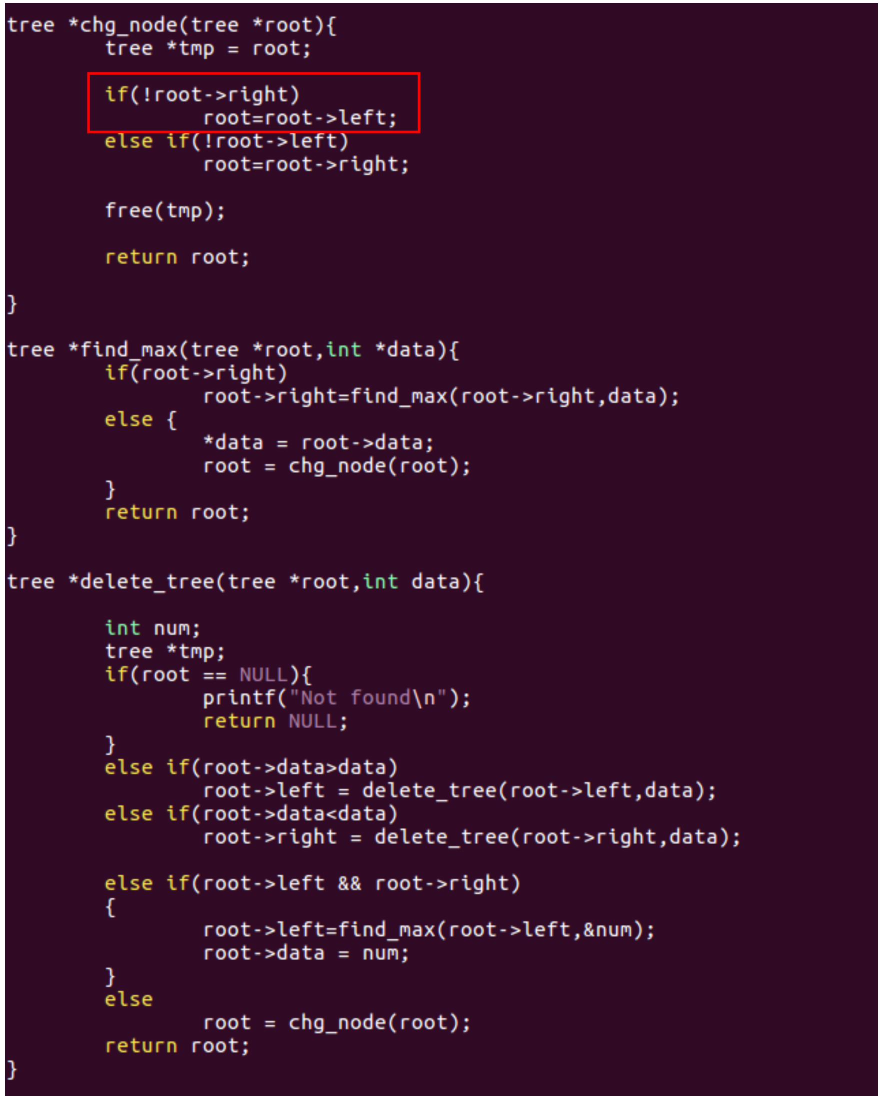
(find_max)

3000 (root) | 100 (data)

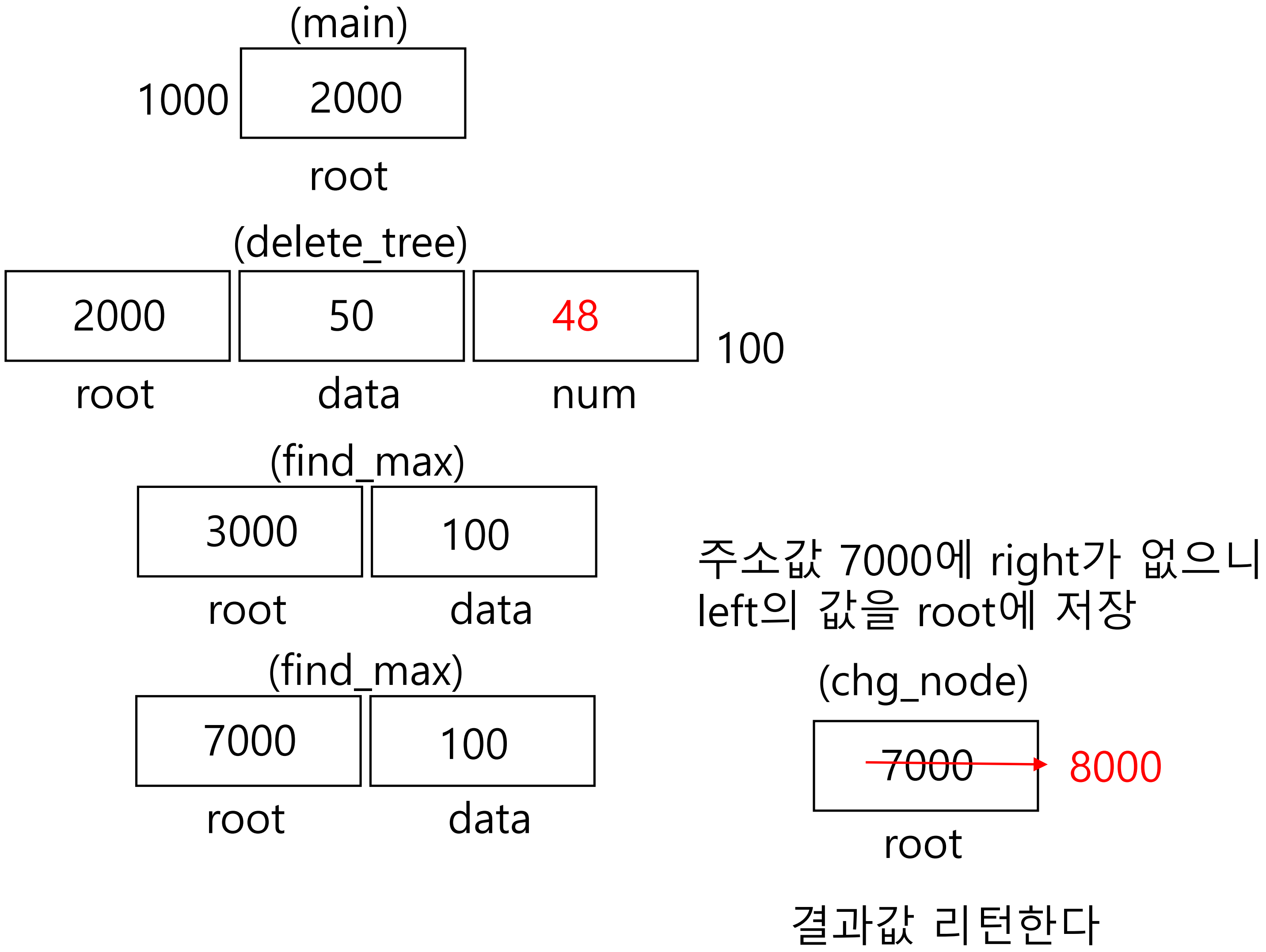
(find_max)

7000 (root) | 100 (data)

주소 7000의
데이터 값



```
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;
        }
        else
                root = chg_node(root);
        return root;
}
```



(main)

1000 2000

root

(delete_tree)

2000 50 48 100

root data num

(find_max)

3000 100

root data

(find_max)

7000 100

root data

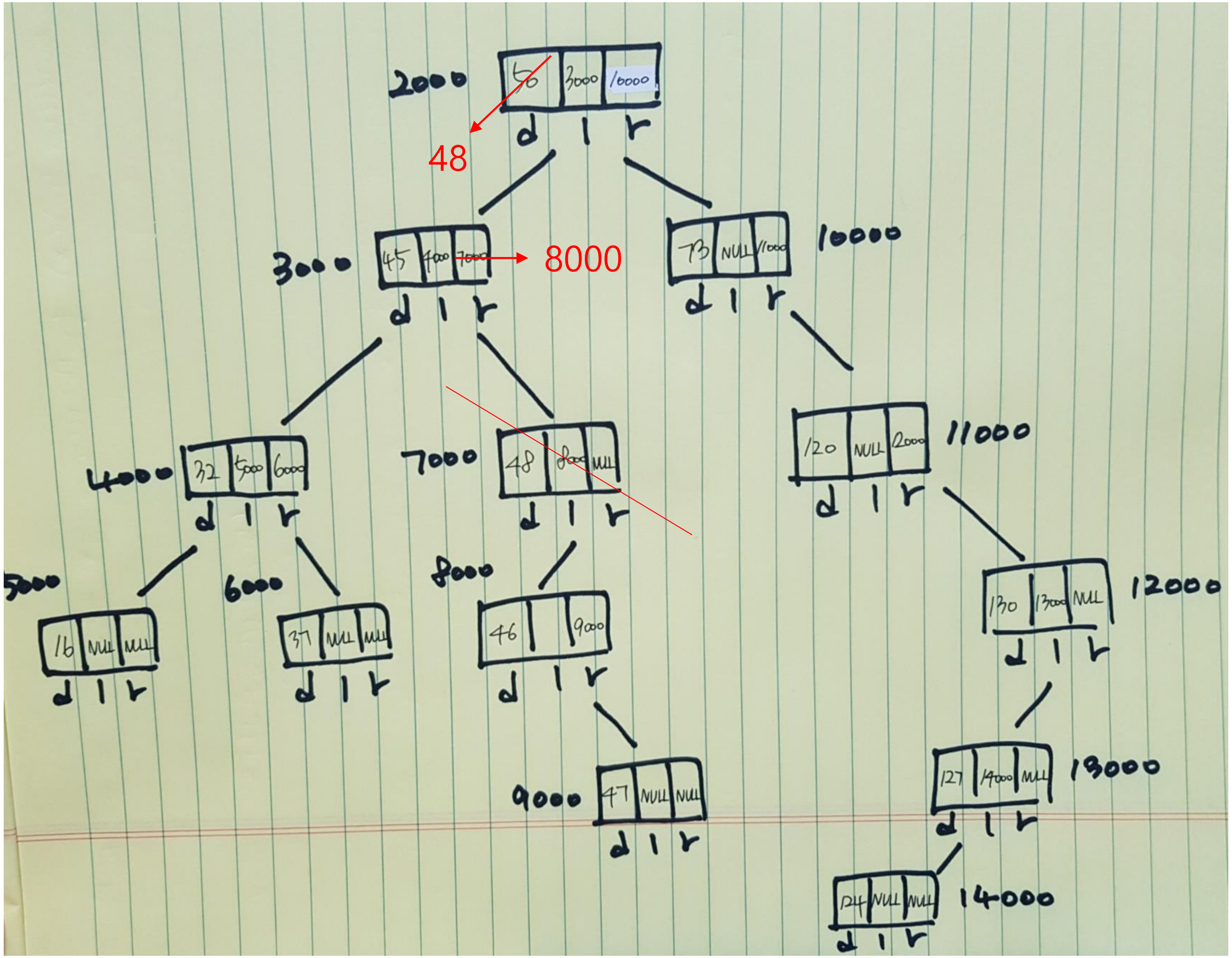
주소값 7000에 right가 없으니
left의 값을 root에 저장

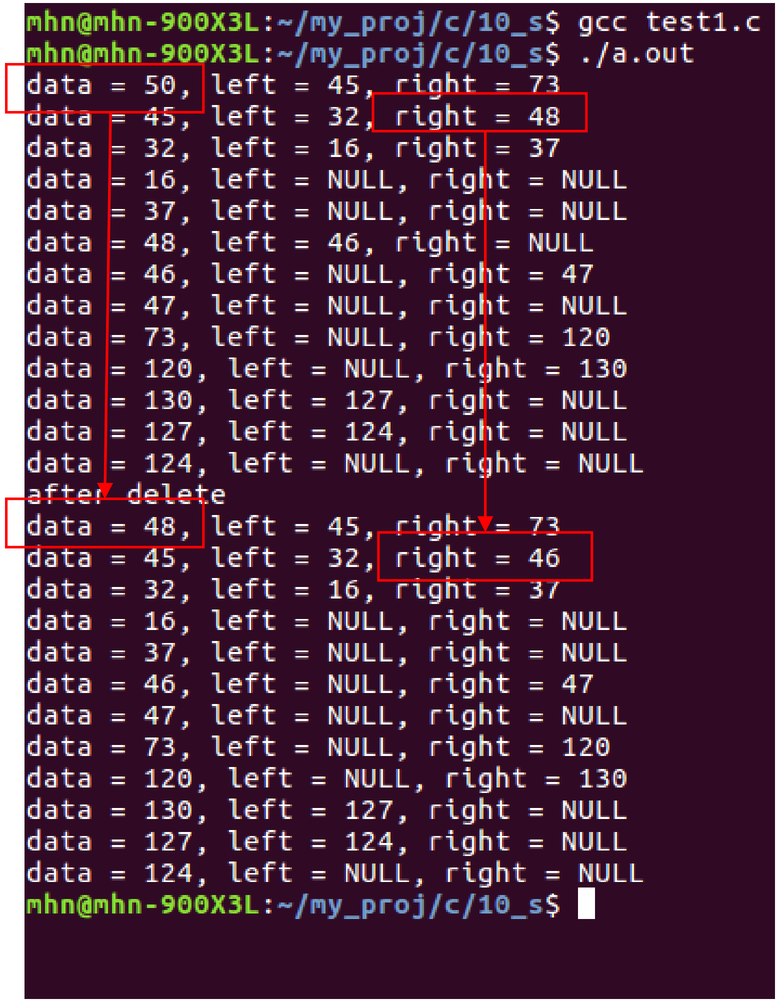
(chg_node)

~~7000~~ 8000

root

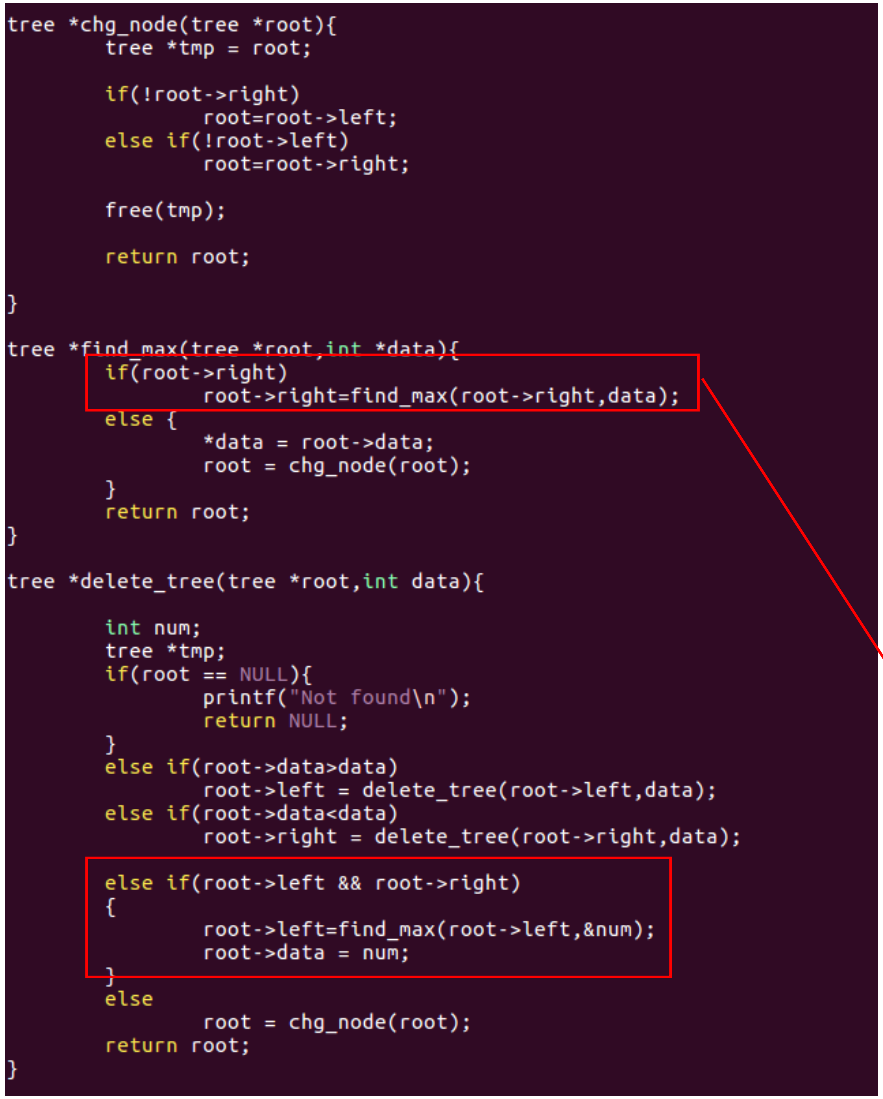
결과값 리턴한다





```
mhn@mhn-900X3L:~/my_proj/c/10_s$ gcc test1.c
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 = 48, left = 45, right = 73
data = 45, left = 32, right = 46
data = 32, left = 16, right = 37
data = 16, left = NULL, right = NULL
data = 37, left = NULL, 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
mhn@mhn-900X3L:~/my_proj/c/10_s$
```

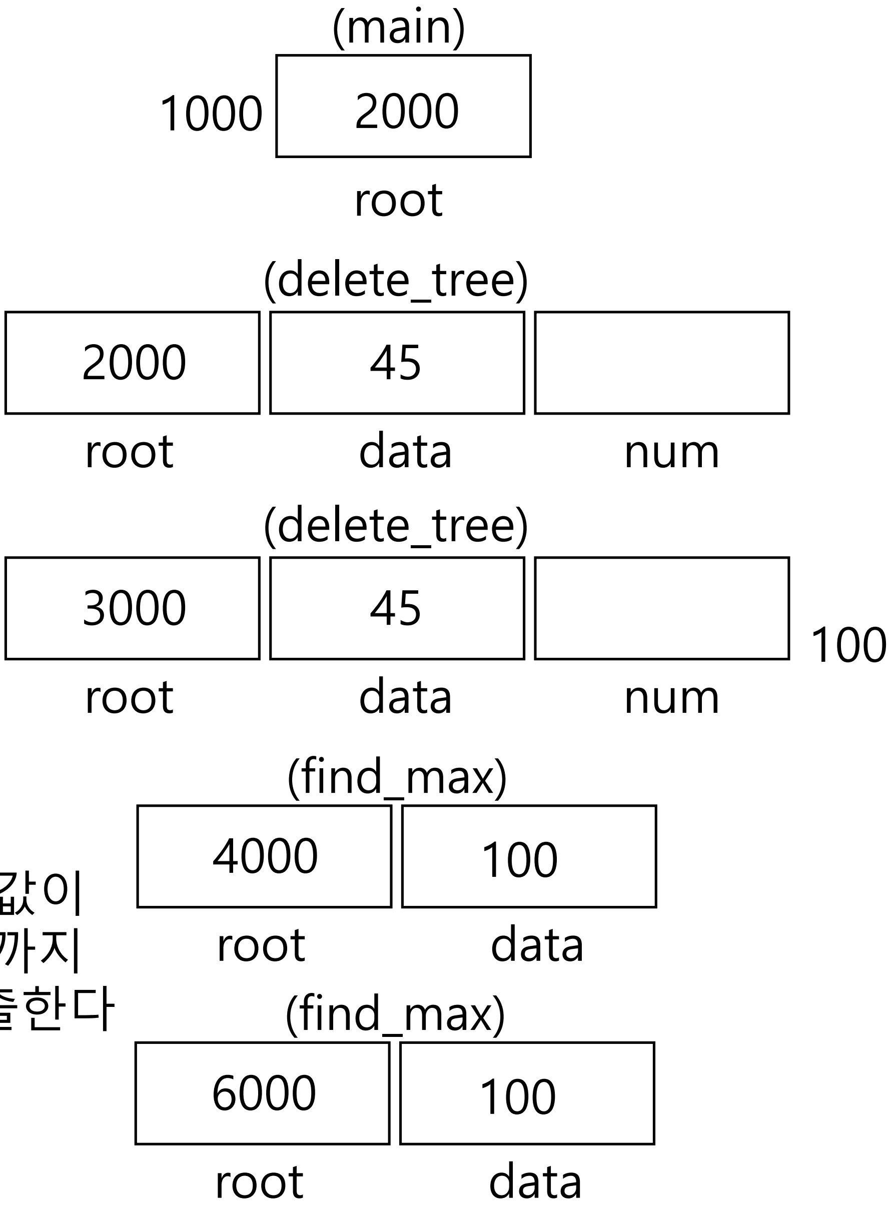
2. Delete 45



```
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;
        }
        else
                root = chg_node(root);
        return root;
}
```

Right의 값이
없을 때 까지
계속 호출한다

left와 right값
둘 다 있을 때
find_max호출



(main)

1000 2000

root

(delete_tree)

2000 45

root data num

(delete_tree)

3000 45 100

root data num

(find_max)

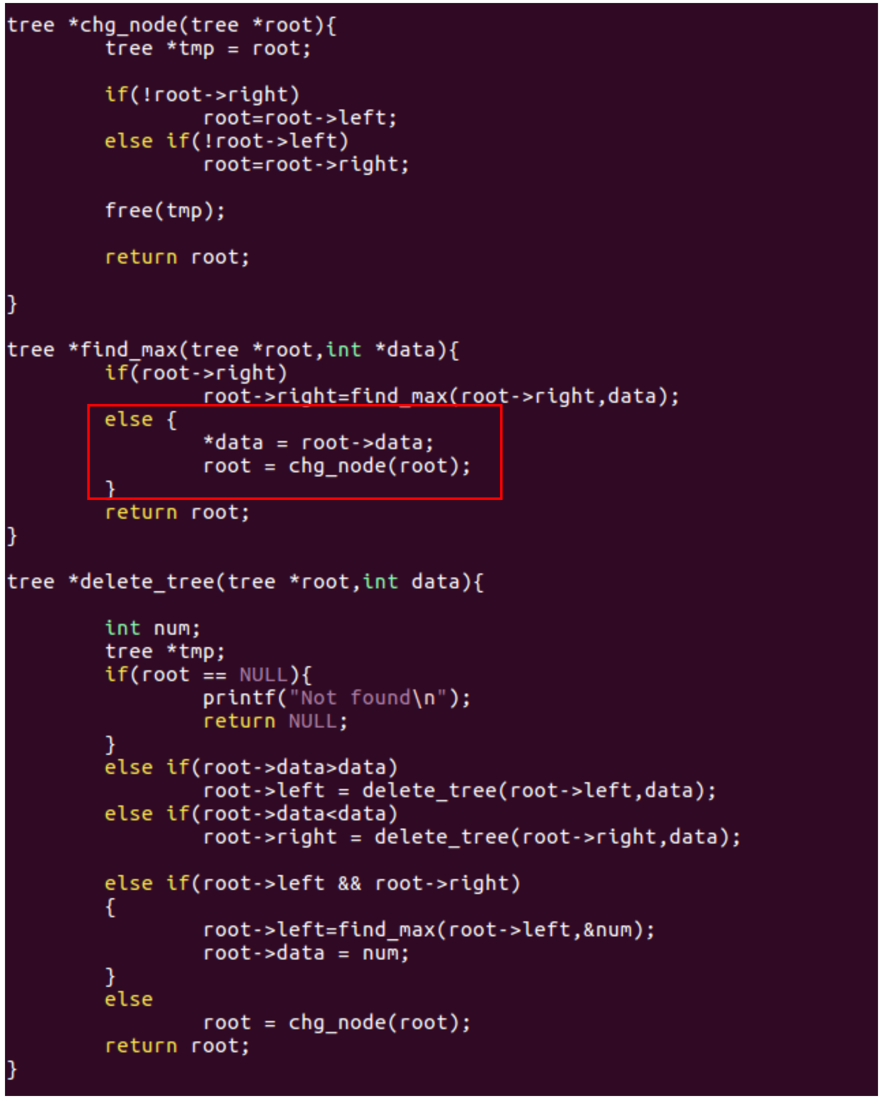
4000 100

root data

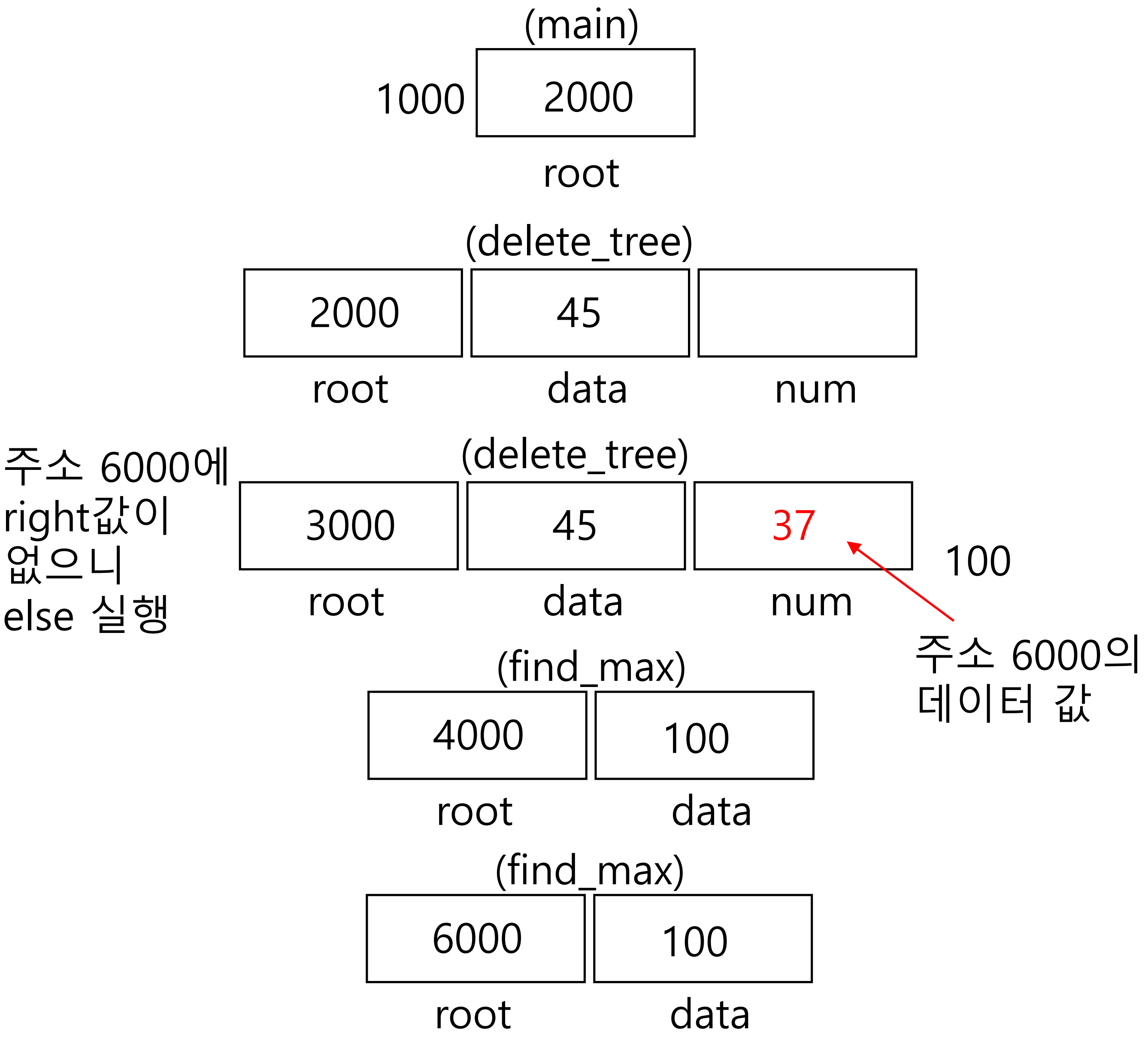
(find_max)

6000 100

root data



```
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;
        }
        else
                root = chg_node(root);
        return root;
}
```



(main)

1000 | 2000

root

(delete_tree)

2000	45	
root	data	num

주소 6000에 right값이 없으니 else 실행

(delete_tree)

3000	45	37
root	data	num

100

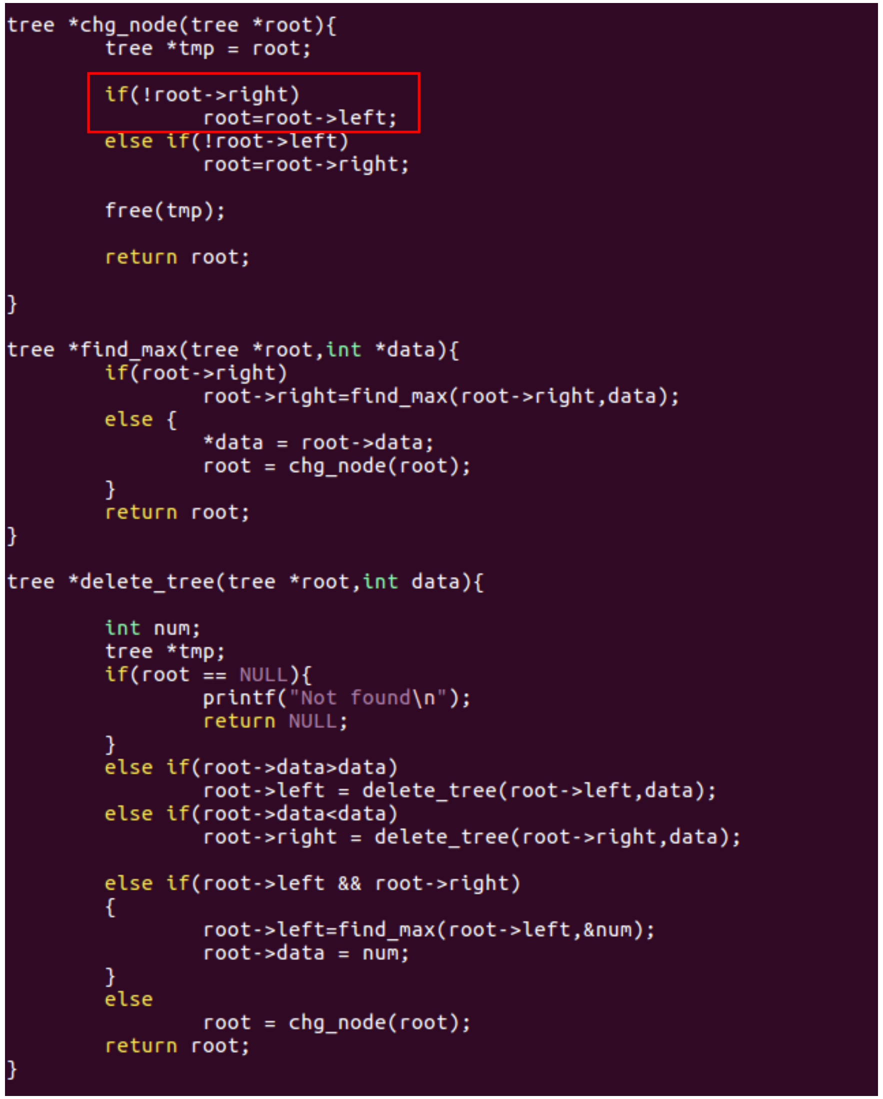
주소 6000의 데이터 값

(find_max)

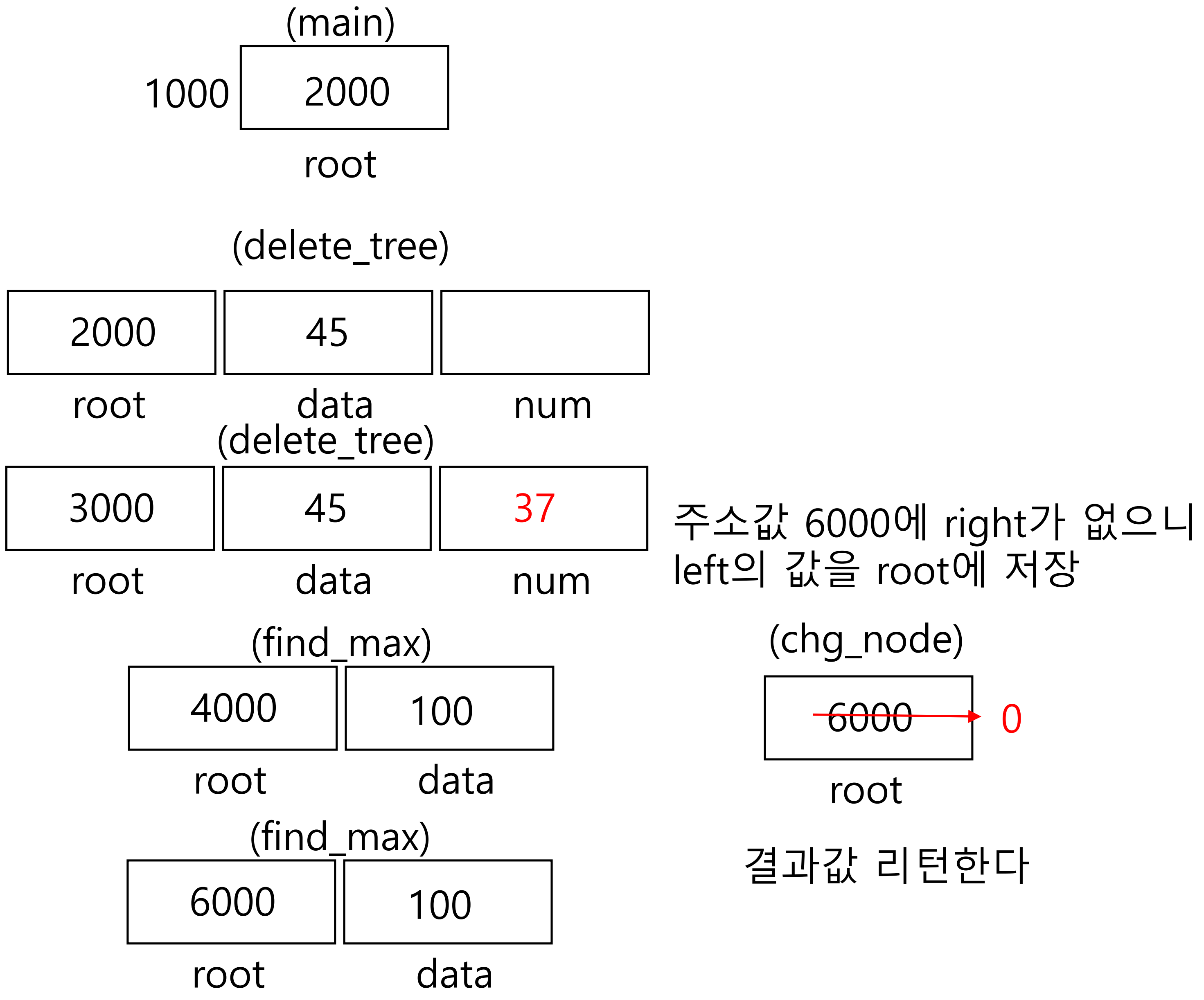
4000	100
root	data

(find_max)

6000	100
root	data



```
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;
        }
        else
                root = chg_node(root);
        return root;
}
```



(main)

1000 | 2000

root

(delete_tree)

2000	45	
root	data	num

(delete_tree)

3000	45	37
root	data	num

(find_max)

4000	100
root	data

(find_max)

6000	100
root	data

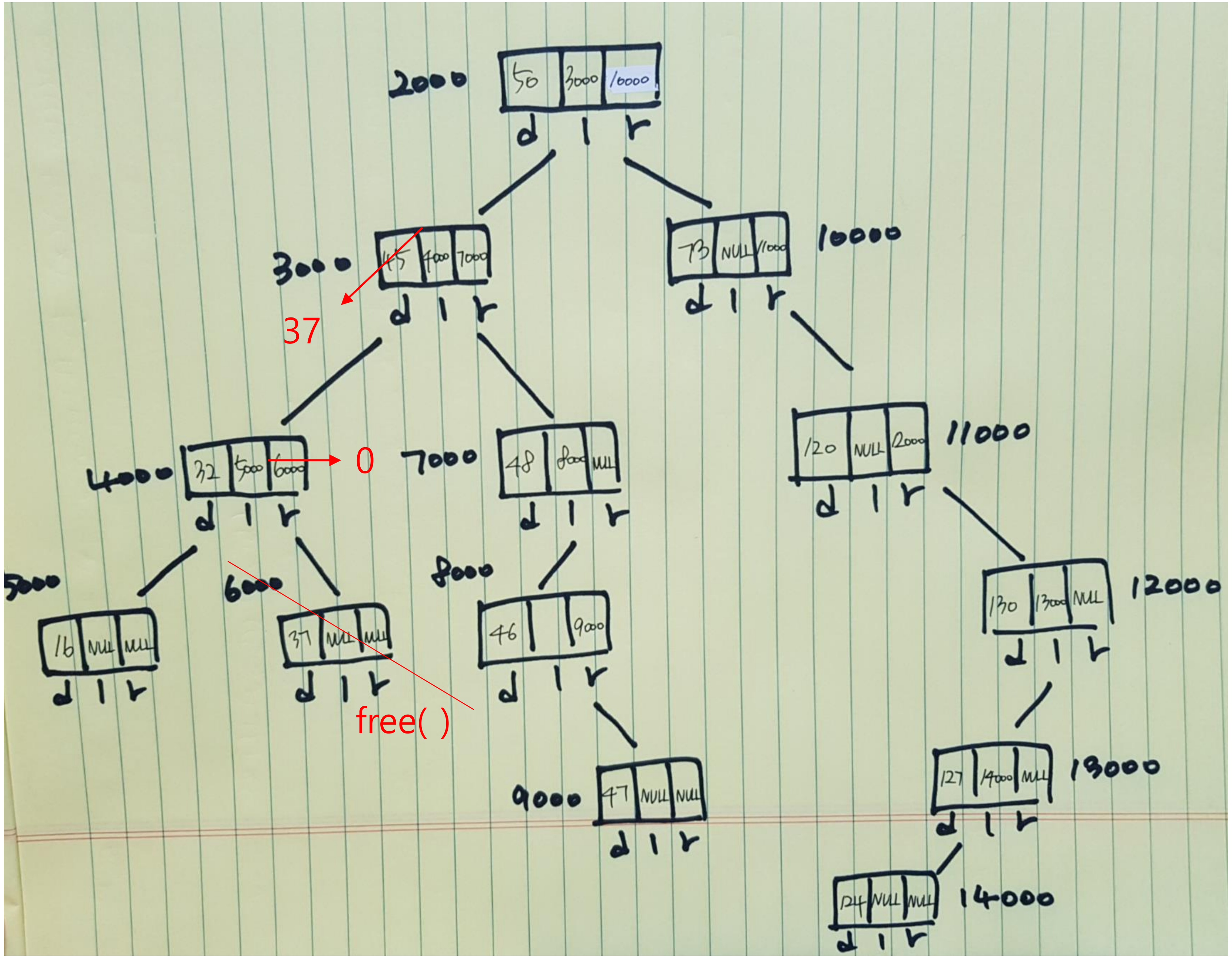
주소값 6000에 right가 없으니
left의 값을 root에 저장

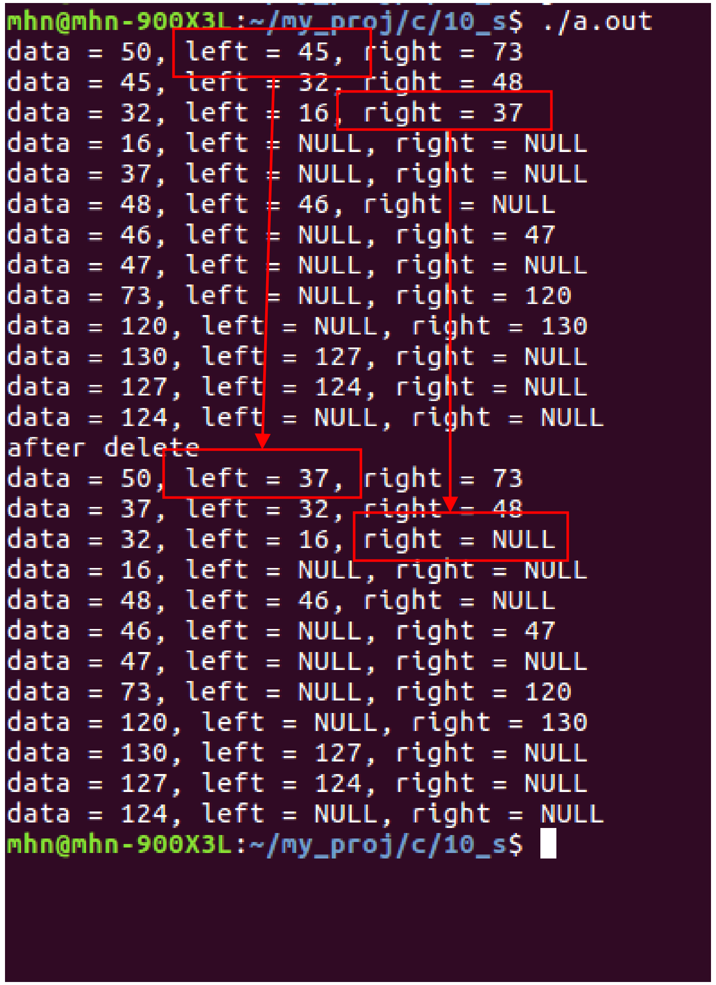
(chg_node)

~~6000~~ → 0

root

결과값 리턴한다





```
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 = 37, right = 73
data = 37, left = 32, right = 48
data = 32, left = 16, right = NULL
data = 16, 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
mhn@mhn-900X3L:~/my_proj/c/10_s$
```