Xilinx Zynq FPGA, TI DSP, MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)
gcccompil3r@gmail.com
학생 - 이유성
dbtjd1102@naver.com

```
#include<stdio.h>
#include<malloc.h>

#define EMPTY 0



struct node{

        int data;
        struct node *left;
        struct node *right;
        };
typedef struct node tree;






tree *get_node()
{
        tree *tmp;
        tmp = (tree *)malloc(sizeof(tree));
        tmp->left =EMPTY;
        tmp->right =EMPTY;
        return tmp;
}









void tree_ins(tree **root, int data)

{
        if(*root ==NULL)
        {
                *root = get_node();
                (*root)->data = data;
                return;
        }
        else if ((*root)->data > data)
                tree_ins(&(*root)->left,data);
        else if ((*root)->data < data)
                tree_ins(&(*root)->right, data);
```

```
}




void print_tree(tree *root)
{
	if(root)
	{
		printf("data = %d, " , root->data);

		if(root->left)
			printf("left = %d, " , root->left->data);
		else
			printf("left = NULL, ");

		if(root->right)
			printf("right = %d\n" , root->right->data);
		else
			printf("right = NULL\n");

		print_tree(root->left);
		print_tree(root->right);
	}
}





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;
}







int main(void)
{

    int i;
    int data[14] = {50, 45, 73 , 32, 48, 46, 16, 37, 120, 47, 130, 127, 124};

    tree *root =NULL;
```

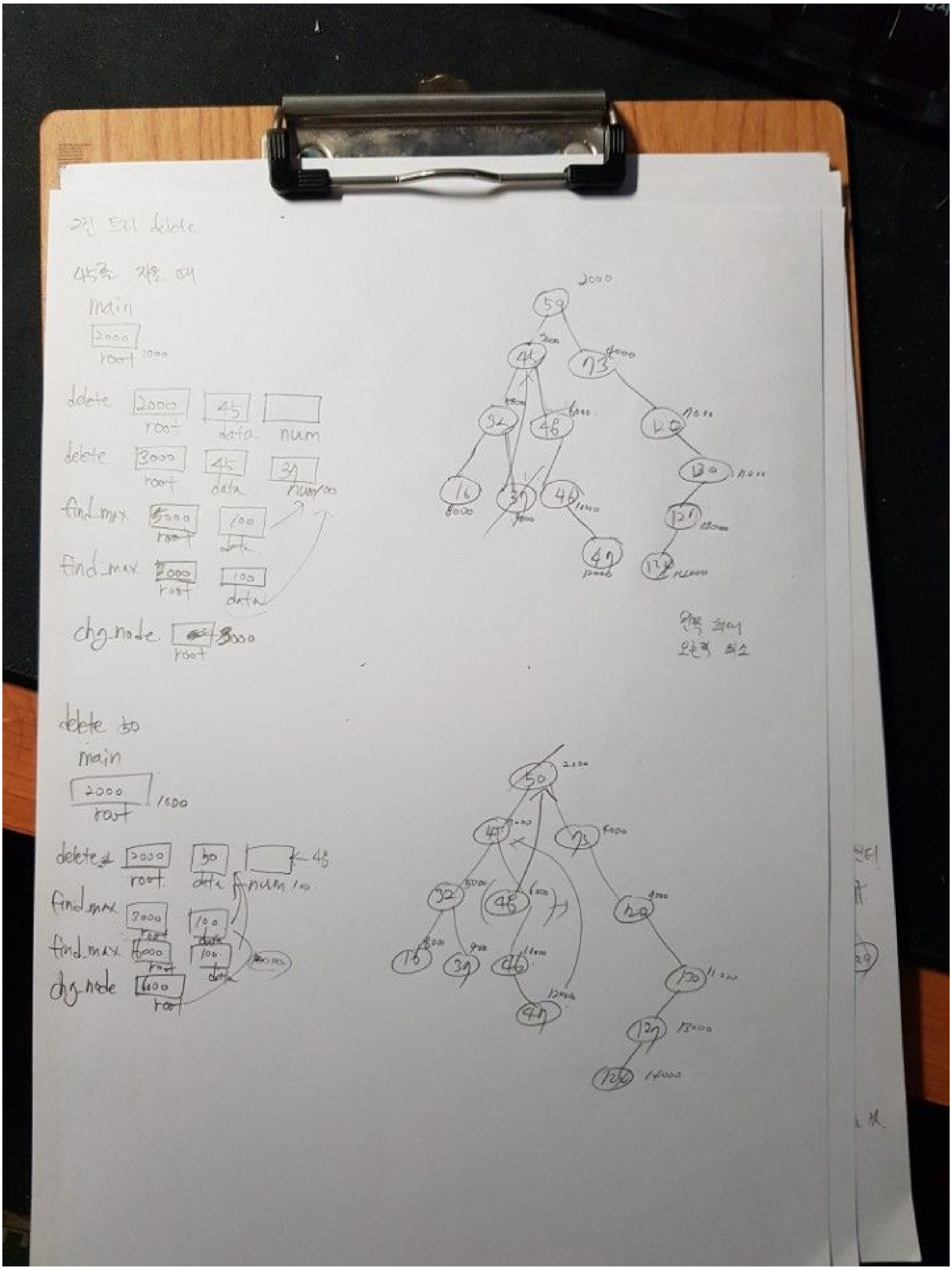
```
        for(i=0; data[i]; i++)
                tree_ins(&root,data[i]);

        print_tree(root);

        delete_tree(root,50);
        printf("After Delete\n");

        print_tree(root);

        return 0;
}
```



이진 트리 delete

45를 지울 때

main

2000
root 1000

delete 2000 root 45 data num

delete 3000 root 45 data 39 num 100

find_max 5000 root 100 data

find_max 2000 root 100 data

chg_node ~~6000~~ 3000 root

왼쪽 최대
오른쪽 최소

delete 50

main

2000
root 1000

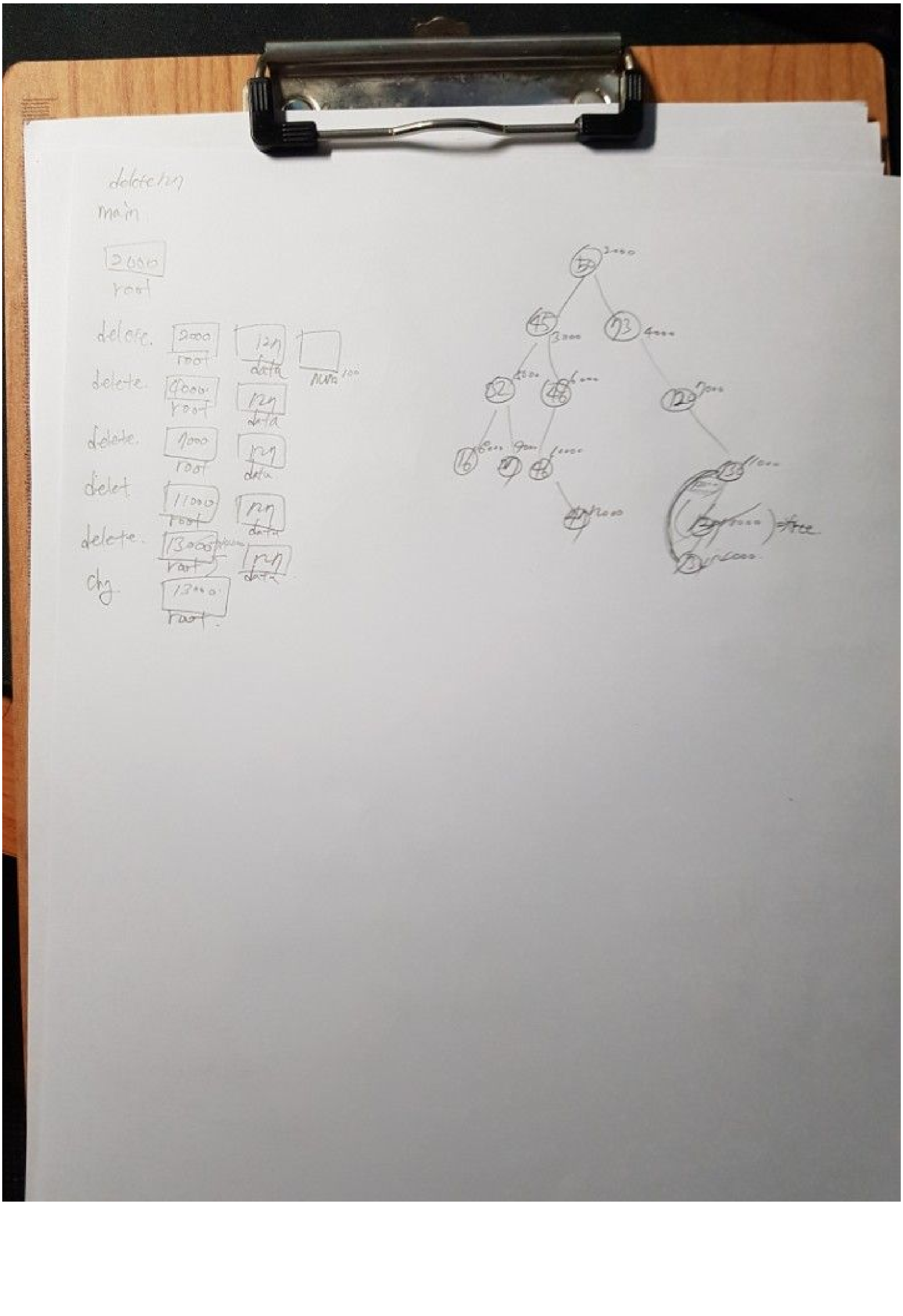
delete 2000 root 50 data num 100 ← 48

find_max 3000 root 100 data

find_max 6000 root 100 data

chg_node 6000 root

50
2000
45
3000
73
4000
32
5000
48
6000
120
7000
16
8000
39
9000
46
11000
130
11000
47
12000
127
13000
126
14000



delete 124

main

2000

root

delete. 2000 root 124 data num 100

delete. 9000 root 124 data

delete. 7000 root 124 data

delet 11000 root 124 data

delete. 13000 root 124 data

chg. 13000 root.

50 2000
45 3000
73 4000
32 5000
48 6000
120 7000
16 8000
9000
10000
11000
12000
13000
free