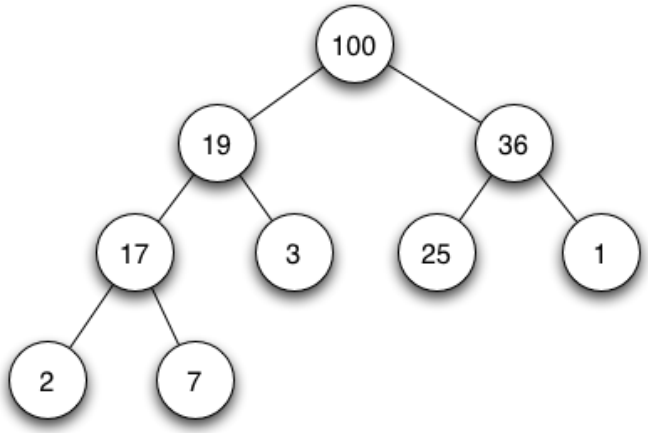
Xilinx Zynq FPGA,TI DSP,
MCU 기반의
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

<2 진트리 딜리트>

1. 왼쪽을 딜리트 할 시는 최대만 올릴 수 있다.(왼쪽이 작다의 트리라서)
2. 오른쪽을 딜리트 시 최소만 올릴 수 있다.(왼쪽이 크다의 트리라서)
3. 최대로 할 시 _1 왼쪽 link 로 한번간다.
   _2 왼쪽 밑으로 한번간 후 오른쪽 끝에 나오는 수이다.




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


#define EMPTY 0

typedef struct __tree
{
	int data;
	struct __tree *link_right;
	struct __tree *link_left;

}tree;

tree *get_node()
{
	tree *tmp;
	tmp = (tree*)malloc(sizeof(tree));
	(tmp)->link_right = EMPTY;
	(tmp)->link_left = EMPTY;
	return tmp;
}
tree *chg_node(tree *root)
{
	tree *tmp = root;

	if(!(root->link_left)){
			free(tmp);
			return root->link_right;
	}

	else if(!(root->link_right)){
			free(tmp);
			return root->link_left;
	}
```

```
}

tree *find_max(tree *root, int *data)
{
/*
	// 값을 찾는다 왼쪽 오른쪽 주소값이 있을 때 한번더 서브로 들어간다.
	if(root->link_right && root ->link_left){
			root = find_max(root->link_right, *num);
			return root;
	}
	// 최대값은 오른쪽에 위치 하므로 오른쪽을 보고 없을때 그 값을 반환하면 최대값이다.
	else if(!(root->link_right)){
			*num = tmp ->data;
			return root;
	}
	*/
	if(root->link_right)
			root->link_right = find_max(root->link_right, data);
	else
	{
			*data = root -> data;
			root = chg_node(root);
	}
	return root;

}

void binary(tree **root, int data)
{
	tree *tmp = *root;
	if(tmp == NULL){
			*root = get_node();
			(*root)->data = data;
			return ;
	}
	if((*root)->data > data)
	{
			 binary(&(*root)->link_left, data);
	}
	else if((*root)->data < data)
	{
			 binary(&(*root)->link_right, data);
	}

}
void print(tree *root)
{
	tree *tmp =root;
	if(root)
```

```
	{
		printf("data = %d ,", tmp->data);
		if(root->link_left){
			printf("left = %d, " ,root->link_left->data);
		}
		else
			printf("left = NUll, ");
		if(root->link_right){
				printf("right = %d\n", root->link_right->data);
		}
		else
				printf("right = NULL\n");


		print(root->link_left);
		print(root->link_right);
	}

}
tree *debinary(tree *root , int data)
{
	int num;
	tree *tmp;
	if(root == NULL)
	{
		printf("Not Found\n");
		return NULL;
	}

	else if(root->data > data)
			root->link_left = debinary(root->link_left, data);
	else if(root->data < data)
			root->link_right = debinary(root->link_right, data);

	else if(root -> link_left && root->link_right)
	{
			root -> link_left = find_max(root->link_left, &num);
			root -> data = num;
	}
	else
			root = chg_node(root);
	return root;
}

/*
tree *debinary(tree *root , int data)
{

	tree *tmp = root;
```

```
	if(tmp == NULL)
	{
			printf("값이 없습니다.\n");
			return NULL;
	}
	if(root->data != data){
		if(root->data > data){

			root->link_left = debinary(root->link_left, data);
		}
		else if(root ->data <data){

			root->link_right = debinary(root->link_right, data);
		}
	}

	else{
		if(root->data > data){
				printf("Now delete %d", data);
				free(tmp);
				return root->link_left;
		}

		else if(root->data < data){
				printf("Now delete %d", data);
				free(tmp);
				return root->link_right;
		}
	}

	return root;

}*/


int main(void)
{
	tree *root=EMPTY;
	int a[]={50, 45, 73, 32, 48, 46, 16, 37, 120, 47, 130 ,127, 124};
	int i, num=0;
	int len = sizeof(a)/sizeof(int);

	printf("len = %d \n",len);

	for(i =0;i<len;i++)
	{
		binary(&root,a[i]);
	}
	print(root);
```

```
    root = debinary(root, 50);
    print(root);
    return 0;
}
```