

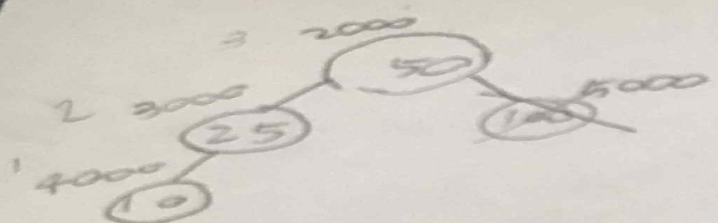
**Xilinx Zynq FPGA, TI DSP, MCU 기반의
프로그래밍 및 회로 설계 전문가 과정
#15**

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1. 배운내용 복습

1) avltree _ delete



main
[2500]
root 1000

ave_del
[1000] [100]
root data

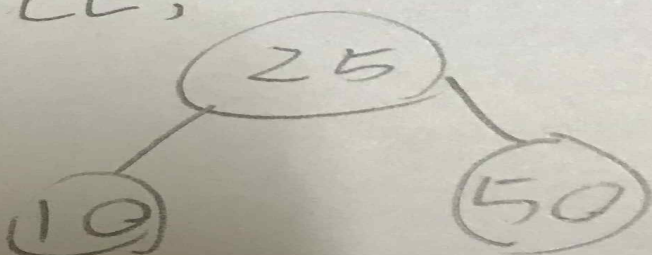
push
[1000]
100

push
[2012] [100]
200

2012
[1000]
tmp

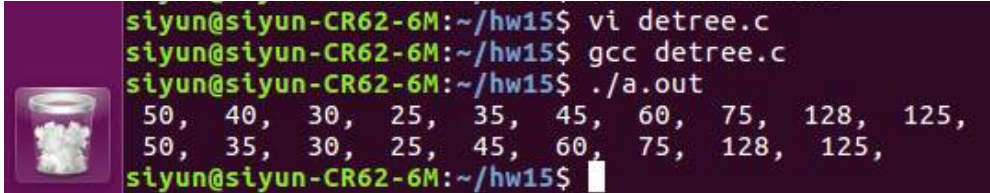
100
[]
top.

int counter=cnt;
chz_node
lev=rot
abs lev > 1
kinds rotation
LL,



2) delete tree

저번에는 tree까지 소스코드를 구현하였다. 오늘은 delete tree를 다시 복습하였다.



```
siyun@siyun-CR62-6M:~/hw15$ vi detree.c
siyun@siyun-CR62-6M:~/hw15$ gcc detree.c
siyun@siyun-CR62-6M:~/hw15$ ./a.out
50, 40, 30, 25, 35, 45, 60, 75, 128, 125,
50, 35, 30, 25, 45, 60, 75, 128, 125,
siyun@siyun-CR62-6M:~/hw15$
```

출력화면

전위로 읽은 방식이며, 위에는 delete를 실행하지 않았을때의 tree, 아래는 delete를 실행했을 경우의 tree이다.

```
tree *detree(tree *root,int data)
{
    int num;
    tree *tmp;
    tmp=root;

    if(root==NULL)
    {
        printf("트리안에 데이터 없음");
        return NULL;
    }
    else if(root->data >data)
    {
        root->left= detree(root->left,data);
    }
    else if(root->data <data)
    {
        root->right = detree(root->right,data);
    }
    else if(root->left && root->right)
    {
        root->left = find_max(root->left,&num);
        root->data=num;
    }else
    {
        chg_node(root);
    }

    return root;
}
```

delete tree의 소스코드이다.

소스코드 형태를 그림으로 그려 이해하고

생각해서 짠 소스코드지만 전에 수업시간해 했던것과 완전 복사본으로 나왔다.

3)insert_tree 비재귀 구현

오늘 하루종일 비재귀를 공부하였는데 아직 delete tree 까지는 무리가있어보인다. delete tree는 함수가 서로 연결되어있어 하나를 바꾸고 확인을 할 수가 없었다.

그래서 먼저 insert tree를 해보았다.

재귀를 통해 반복하던걸 비재귀일때는 함수가 아닌 while문으로 대체한다.

여기서

```
root=&(*root)->left;
```

```
(*root)->left=find_max(&(*root)->left,data);
```

와 같은 개념들이 정말 중요한거같다.

더블포인터의 주소,데이터 접근 형태가 많이 어려웠었다.

```
siyun@siyun-CR62-6M:~/hw15$ gcc fastdetree.c
siyun@siyun-CR62-6M:~/hw15$ ./a.out
비재귀 인설프트리
50, 40, 30, 25, 35, 45, 60, 75, 128, 125,
after delete tree
50, 35, 30, 25, 45, 60, 75, 128, 125,
siyun@siyun-CR62-6M:~/hw15$
```

```
void insert_tree2(tree **root,int data)
{
    tree **tmp=root;
    while(*tmp)
    {
        if((*tmp)->data > data)
        {
            tmp=&(*tmp)->left;
        }
        else if((*tmp)->data < data)
        {
            tmp = &(*tmp)->right;
        }
    }
    *tmp=get_node();
    (*tmp)->data=data;
}

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

4) 4096 단위.

정말 억울해서 되게 생각해봤다.

처음에는 4096에다 &연산을 했었고

a와 ~a를 or 연산한 후 4096과 and 연산도 해보았지만 실패하였다.

하지만 4095로 해결하였다.

4095를 이진수로 바꾸면

0111 1111 1111 이다

여기서 0은 부호비트

8000을 2진수로 바꾸면

1XXX XXXX XXXX 일 것이다.

그러면 4096단위로 나오기위해서는

4095를 NOT붙여 반전시키고 AND를 시키면 답이 나온다.

```
siyun@siyun-CR62-6M:~/hw15$ vi bit.c
siyun@siyun-CR62-6M:~/hw15$ gcc bit.c
siyun@siyun-CR62-6M:~/hw15$ ./a.out
8000 이상을 입력해보시오 비트단위 읽기 : 8000

입력한 값 8000 = 4096
siyun@siyun-CR62-6M:~/hw15$ ./a.out
8000 이상을 입력해보시오 비트단위 읽기 : 9000

입력한 값 9000 = 8192
siyun@siyun-CR62-6M:~/hw15$ ./a.out
8000 이상을 입력해보시오 비트단위 읽기 : 10000

입력한 값 10000 = 8192
siyun@siyun-CR62-6M:~/hw15$ ./a.out
8000 이상을 입력해보시오 비트단위 읽기 : 11000

입력한 값 11000 = 8192
siyun@siyun-CR62-6M:~/hw15$ ./a.out
8000 이상을 입력해보시오 비트단위 읽기 : 13000

입력한 값 13000 = 12288
siyun@siyun-CR62-6M:~/hw15$
```

```
#include <stdio.h>

int main(void)
{
    int a;
    int b=4095;

    printf("8000 이상을 입력해보시오 비트단위 읽기 : ");
    scanf("%d",&a);
    printf("\n");

    printf("입력한 값 %d = %d \n",a,a&(~b));

    return 0;
}
```