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

#9

강사 : Innova Lee(이 상훈)

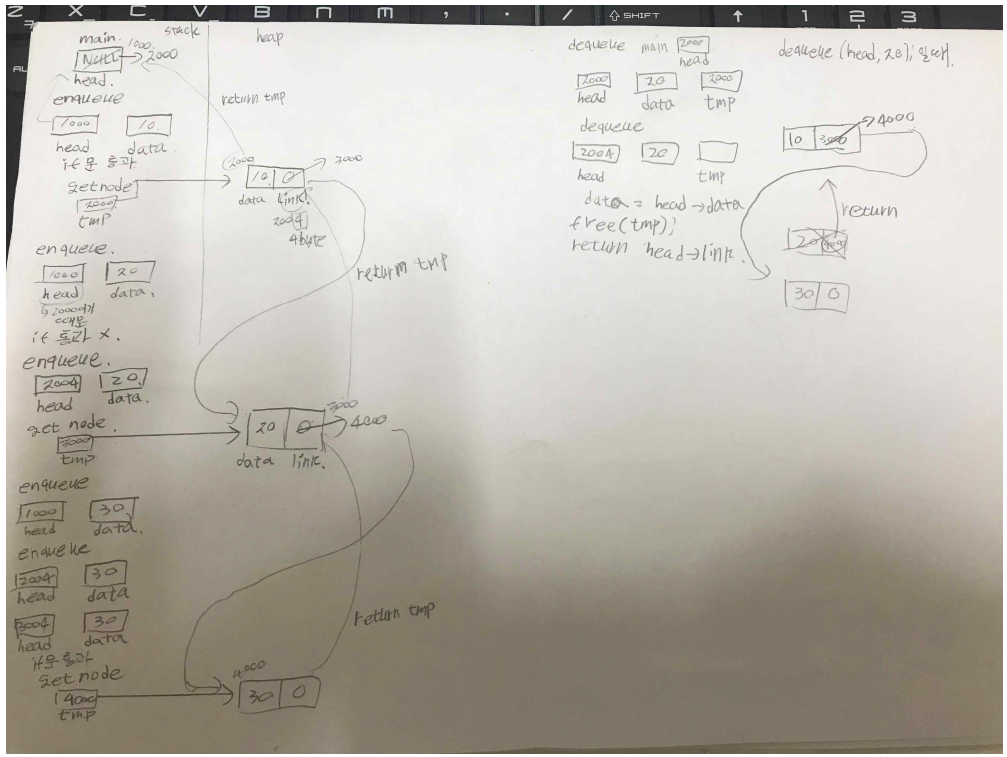
학생 : 김 시윤

1. 배운내용 복습

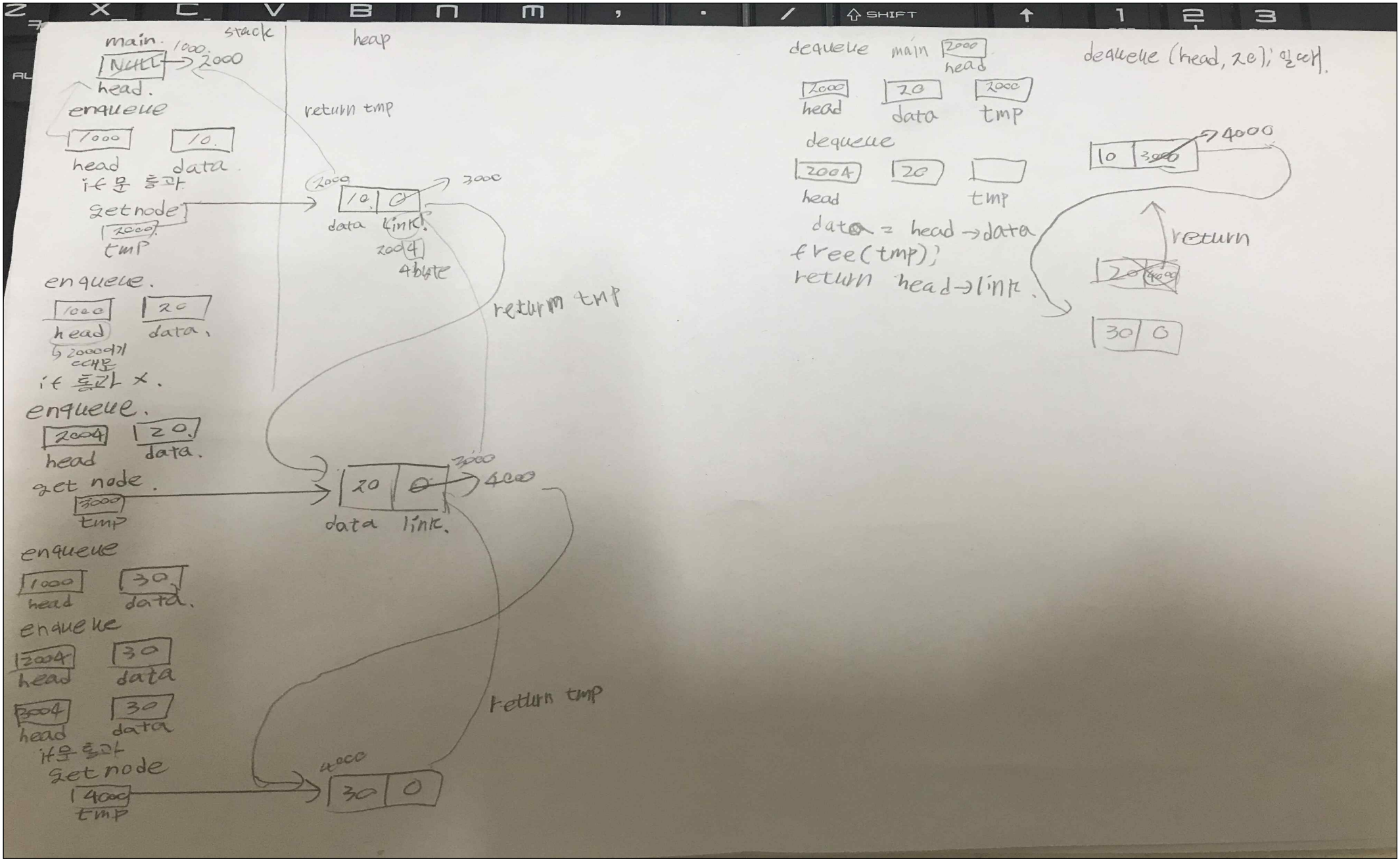
1) deQueue

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

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};
typedef struct node queue;

queue *get_node()
{
        queue *tmp;
        tmp = (queue *)malloc(sizeof(queue));
        tmp->link=EMPTY;
        return tmp;
}

void enqueue(queue **head,int data)
{
        if(*head ==NULL)
        {
                *head = get_node();
                (*head)->data = data;
                return;
        }
        enqueue(&(*head)->link,data);
}
```

```
queue *dequeue(queue *head,int data)
{
        queue *tmp = head;

        if(tmp == NULL)
        {
        printf("There are no data that you delete₩n");
        }

        if(head->data != data)
        {
         head->link = dequeue(head->link, data);
        }
        else
        {
                //queue *res = head->link;
                printf("Now you delete %d₩n", data);
                free(tmp);
         }
}
int main(void)
{
        queue *head = EMPTY;
        enqueue(&head,10);
        enqueue(&head,20);
        enqueue(&head,30);
        dequeue(head,20);
        print_queue(head);

        return 0;
}
```



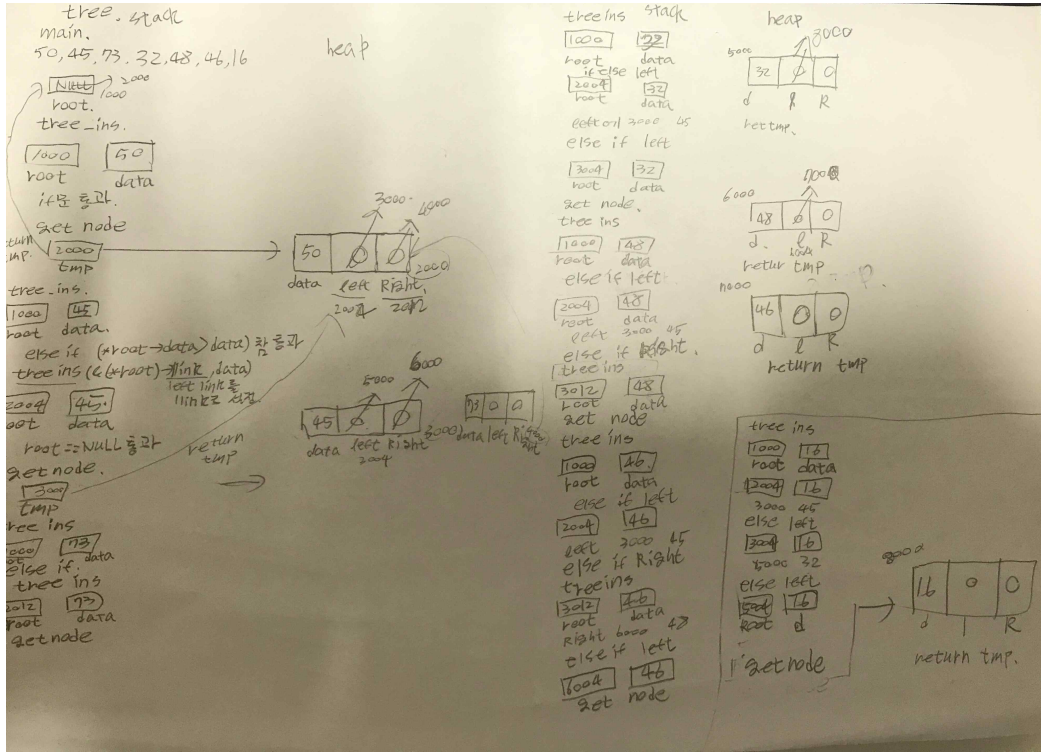
deQueue 사진 확대



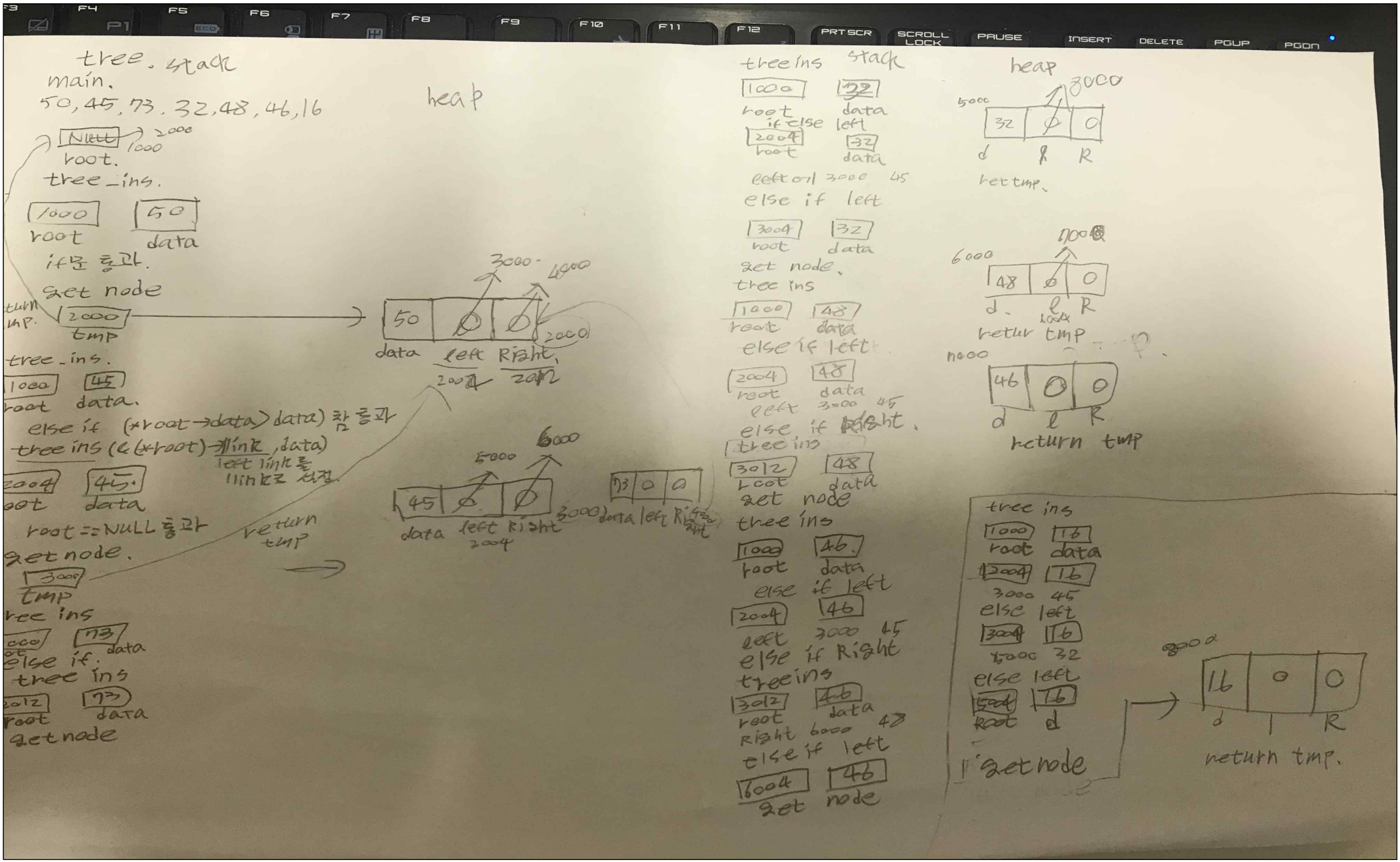
2) tree 7개 값까지.

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#define EMPTY 0

struct node{
        int data;
        struct node *rlink;
        struct node *llink;
};
typedef struct node tree;

tree *get_node()
{
        tree *tmp;
        tmp = (tree *)malloc(sizeof(tree));

        tmp->llink=EMPTY;
        tmp->rlink=EMPTY;
        return tmp;
}
void insert(tree **root,int data)
{
        if(*root ==NULL)
        {
                *root = get_node();
                (*root)->data = data;
                return;
        }
        if((*root)->data > data)
        {
          insert(&(*root)->llink,data);
        }
        else
        {
        insert(&(*root)->rlink,data);}
}
```

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

                  else
                          printf("left = NULL, ");
              if(root->rlink)
              printf("right = %d₩n", root->rlink->data);
              else
              printf("right = NULL₩n");
              print_tree(root->llink);
              print_tree(root->rlink);
        }
int main(void)
{
        tree *root = EMPTY;
        int i;
int a[20]={50,45,73,32,48,46,16,37,120,47,130,127,124};
        for(i=0;i<a[i];i++)
{
        insert(&root,a[i]);
}
        print_tree(root);

        return 0;
}
```



tree 사진 확대


tree. stack
main.
50, 45, 73, 32, 48, 46, 16
heap
root.
tree_ins.
root
data
if문 통과.
get node
tmp
data left Right
tree_ins.
root data.
else if (*root->data > data) 참 통과
tree ins (&(*root) link, data)
left link를
link로 전달
root == NULL 통과
get node.
tmp
tree ins
else if.
tree ins
root
get node
return
tmp
tree ins stack
root
if else
data
left
get node.
tree ins
else if left
else if Right
heap
ret tmp.
return tmp
return tmp.
tree ins
root
data
else left
else if Right
tree ins
root
Right
else if left
get node
return tmp.

print tree 설명.

위의 그림을 보면서 설명하면.

```
 if(root)
          {
            printf("data  =  %d,  ",root->data);
                 print_tree(root->llink);
                      print_tree(root->rlink);
          }
```

root 가 2000이기 때문에 if문을 통과한다.
root 의 첫 값은 50을 뿌려준다.
50을 기준으로 link left 에 있는값들을 뿌려주고
리턴 하면서 right 에 값이 있으면 뿌려준다.