자료구조 HW3

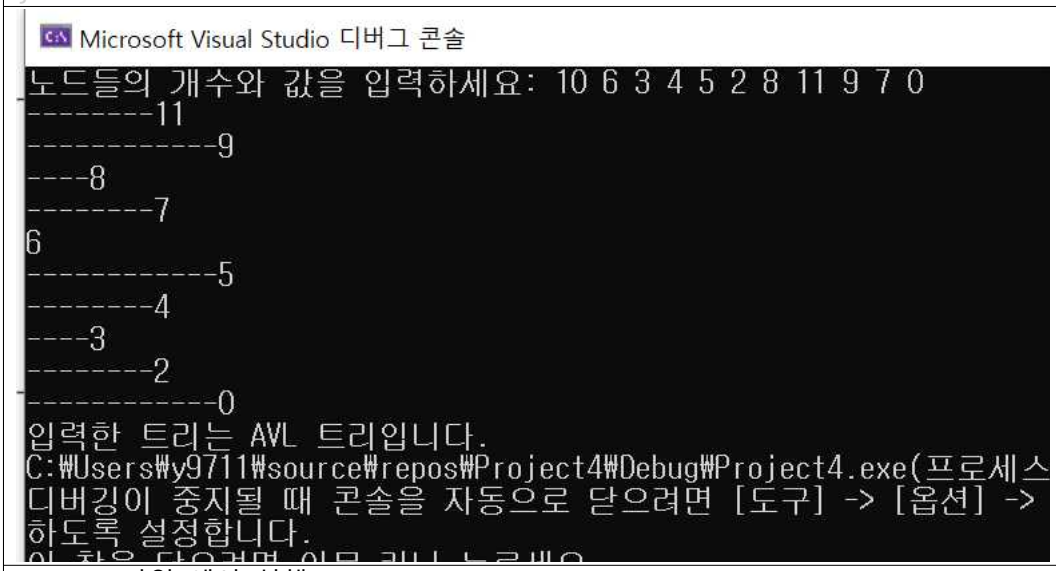
2016124145 양해찬

1.

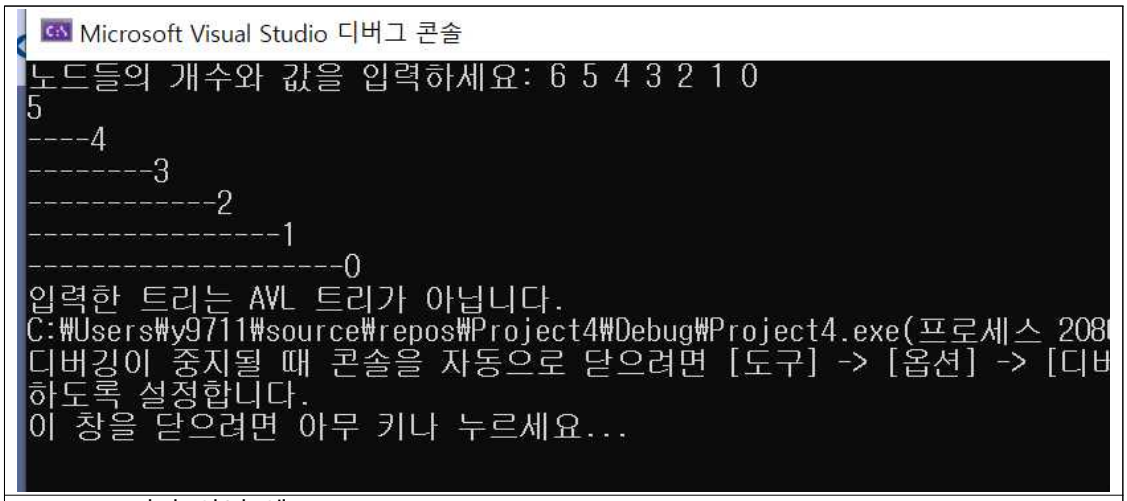
```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
typedef struct _node {
    int key;
    struct _node* left;
    struct _node* right;
} node;
void init_tree(node** p) {
    *p = (node*)malloc(sizeof(node));
    (*p)->left = NULL;
    (*p)->right = NULL;
}
node* bti_insert(int key, node* base, int* num) {
    node* p, * s;
    p = base;
    s = base->left;
    while (s != NULL) {
        p = s;
        if (key < s->key) s = s->left;
        else s = s->right;
    }
    if ((s = (node*)malloc(sizeof(node))) == NULL)
        return NULL;
    s->key = key;
    s->left = NULL;
    s->right = NULL;
    if (key < p->key || p == base) p->left = s;
    else
        p->right = s;
    (*num)++;
    return s;
}
int get_level(node* s) { // 트리의 level을 구함
    if (s == NULL)
        return 0;
    else {
        int left_l = get_level(s->left); // 왼쪽 노드의 level구함
```

```
        int right_l = get_level(s->right); // 오른쪽 노드의 level구함
        return 1 + (left_l > right_l ? left_l : right_l); // 큰 값에 1을
더해서
    }
}
int print_node(node* s, int level) {
    if (s != NULL) {
        level++;                            //호출 될 때 마다 level++
        print_node(s->right, level);        //오른쪽에 대해서 print
        for (int i = 0;i < level - 1;i++) printf("----");   //level 만큼
----출력
        printf("%d\n", s->key);                              //key출력
        print_node(s->left, level);                         //왼쪽에 대해서도
똑같이 실행
    }
}
int AVL(node* s) {  //avl 여부 확인 함수
    int i = 0;
    if (s != NULL) {
        AVL(s->left);   //왼쪽 서브트리부터
        if ((s->right != NULL) && (s->left != NULL)) {
            if ((get_level(s->left) - get_level(s->right)) > 1 || (get_l
evel(s->left) - get_level(s->right) < -1)) {
                i = 1;          //좌측과 우측의 레벨을 각각 구하고 빼서 avl인
지 확인하고 아니면 1을 return한다.
                return i;
            }
        }
        else if (((s->left == NULL) && (s->right != NULL) && (get_level
(s->right) > 1)) || ((s->right == NULL) && (s->left != NULL) && (get_lev
el(s->left) > 1))) {
            i = 1;          //만약 좌측 우측 중 한쪽만 자식 노드가 있는 경우
            return i;
        }
        AVL(s->right);//오른쪽 서브트리도 똑같이 수행
    }
}
int main() {
```

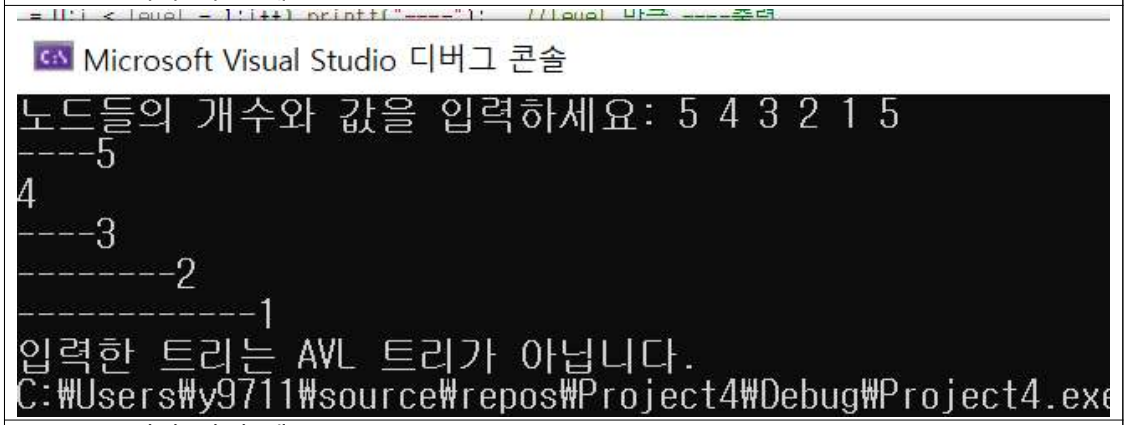
```
    node* tree;
    node* s;
    int i = 0, j, num, level = 0;
    init_tree(&tree);
    printf("노드들의 개수와 값을 입력하세요: ");
    scanf("%d", &num);
    int* n = (int*)malloc(sizeof(int) * num);    //개수만큼 노드 입력받을
공간
    for (j = 0;j < num;j++) {
        scanf("%d", &n[j]);
    }
    while (i != num) {//입력받은 노드들을 insert
        bti_insert(n[i], tree, &i);
    }
    print_node(tree->left, level);//노드 출력
    if (AVL(tree->left) != 1) printf("입력한 트리는 AVL 트리입니다.");
    else            printf("입력한 트리는 AVL 트리가 아닙니다.");
}
```



>> pdf 파일 예시 실행



>> avl트리가 아닌 예 1



>> avl트리가 아닌 예 2