2 번

문제;:

2. (25 points: Programming)
Implement the function that finds a successor of node in the inorder traversal.
   - Refer to the following pseudo code.
   - Use the given data structure and simple binary tree example in p3.

```
Tree_successor(x)
{
        if x->right != NULL //x's right subtree is not null
                return the leftmost node of right subtree

        //x's right subtree is null
        y = x->parent
        while (y != NULL and x == y->right) {
                x = y;
                y = y->parent;
        }
        return y;
}
```

문제해결:

각 노드 구조체에 parent 가 들어가있다. 노드 p 의 오른쪽 링크 노드를 q 에 넣어주고 q 가 null 이 아닐 때 맨 왼쪽에 있는 노드를 찾아서 출력해준다 parent 를 이용하여 node – parent 관계가 left 가 될 때까지 올라가서 left 관계가 되면 parent 를 출력해준다.

실행화면:


C:\Users\ssh12\CLionProjects\untitled30\cmake-build-debu
A
C
B
G
D
F
E
Process finished with exit code 0

3 번 문제:

3. (25 points: Programming)
Implement the function that finds a predecessor of node in the inorder traversal.
   - Write up the pseudo code by referring to the pseudo code of 'Tree_successor(x)'.
   - Use the given data structure and simple binary tree example in p5.

문제해결:

각 노드 구조체에 parent 가 들어가있다. 노드 p 의 왼쪽 링크 노드를 q 에 넣어주고 q 가 null 이 아닐 때 맨 오른쪽에 있는 노드를 찾아서 출력해준다 parent 를 이용하여 node – parent 관계가 right 가 될 때까지 올라가서 right 관계가 되면 parent 를 출력해준다.

실행화면:

```
E
F
D
G
B
C
A

Process finished with exit code 0
```

문제

4 번

4. (40 points: Programming)

In the code provided in the page 60 and 61, the case 3 was implemented using the successor at the right subtree. Revise this part by using the predecessor at the left subtree. Verify your code by running the following three examples. Namely, the revised code should produces the same results of the original code. Refer to the attached original code ('bst_insertion_deletion.cpp').

```
typedef struct TreeNode {
int key;
struct TreeNode *left, *right;
} TreeNode;

//                  35
//          18                  68
//   7             26                  99
//3     12     22         30
//    10          24

TreeNode n12 = { 24, NULL, NULL };
TreeNode n11 = { 10, NULL, NULL };
TreeNode n10 = { 30, NULL, NULL };
TreeNode n9 = { 22, NULL, &n12 };
TreeNode n8 = { 12, &n11, NULL };
TreeNode n7 = { 3,  NULL, NULL };
TreeNode n6 = { 99, NULL, NULL };
TreeNode n5 = { 26, &n9, &n10 };
TreeNode n4 = { 7,  &n7, &n8 };
TreeNode n3 = { 68, NULL, &n6 };
TreeNode n2 = { 18, &n4, &n5 };
TreeNode n1 = { 35, &n2, &n3 };
```

Three examples

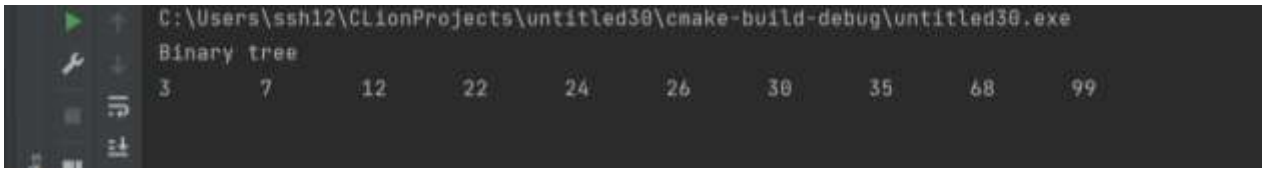
1) key = 18
2) key = 35
3) key = 7

```
void main()
{
        TreeNode *root = &n1;

        printf("Binary tree\n");
        inorder(root);
        printf("\n\n");

        delete_node(&root, key);

        printf("Binary tree\n");
        inorder(root);
        printf("\n\n");
}
```

문제해결:

Subtree 가 2 개가 있는 경우 노드를 삭제할 때 왼쪽에서 큰 값이나 오른쪽에서 가장 작은 값을 successor 로 선택한다. 기존 코드에서 left 를 right 로 right 를 left 로 바꿔서 해결하였다.

실행결과:


C:\Users\ssh12\CLionProjects\untitled30\cmake-build-debug\untitled30.exe
Binary tree
3 7 12 22 24 26 30 35 68 99