Data Structure

Practical Class - Week 10

Exercise01 – Skeleton 1

```c
#pragma warning(disable: 4996)

#include<stdio.h>
#include<stdbool.h>
#include<stdlib.h>
#include<memory.h>
#include<malloc.h>

typedef char element;

struct node;
struct tree;
typedef struct node* node_ptr;
typedef struct tree* tree_ptr;

typedef struct node {
    element key;        // data field
    node_ptr left;      // pointer of the left child node
    node_ptr right;     // pointer of the right child node
    node_ptr parent;    // pointer of the parent node
} node;

typedef struct tree{
    node_ptr root;      // pointer of the root node of tree.
};

tree_ptr make_tree(){
    tree_ptr new_tree = (tree_ptr)malloc(sizeof(struct tree));
    new_tree->root = NULL;
    return new_tree;
}
```

Exercise01 – Skeleton 2

```
33  /**
34   * @brief       해당 노드의 서브 트리를 in-order 순서로 출력해주는 함수
35   *              올바른 BST의 경우 in-order로 출력하면 오름차순으로 정렬되어 출력된다.
36   *              [TIP] 테스트용으로 중간 중간 사용하면 유용하다.
37   *
38   * @param node  탐색과 출력을 수행 할 노드의 포인터
39   */
40  void print_inorder(node_ptr node){
41      if (node->parent == NULL){
42          printf("{ ");
43      }
44
45      if (node != NULL){
46          if (node->left != NULL){
47              print_inorder(node->left);
48              printf(", ");
49          }
50
51          printf("%c", node->key);
52
53          if (node->right != NULL){
54              printf(", ");
55              print_inorder(node->right);
56          }
57      }
58
59      if (node->parent == NULL){
60          printf(" }\n");
61      }
62  }
```

Exercise01 – Problem 1

```
64  /**
65   * @brief   새로운 key값을 가진 노드를 생성한 후 BST에 삽입한다.
66   *          만약, 중복된 key가 이미 존재한다면 에러 메시지를 출력하고 NULL을 반환한다.
67   *
68   * @param tree          노드를 추가 할 트리의 포인터
69   * @param new_key       새로 추가 할 노드의 key 값
70   * @return node_ptr     새로 생성된 노드의 포인터
71   */
72  node_ptr insert(tree_ptr tree, element new_key){
73      node_ptr parent = NULL;
74      node_ptr candidate = tree->root;
75
76      node_ptr new_node = (node_ptr)malloc(sizeof(struct node));
77      new_node->left = NULL;
78      new_node->right = NULL;
79      new_node->parent = NULL;
80      new_node->key = new_key;
81
82      // tree의 루트가 없는 경우 이 노드를 루트로 저장한 후 종료.
83      if (tree->root == NULL){
84          tree->root = new_node;
85          return new_node;
86      }
87
88
89
90
91
92      return new_node;
93  }
```

Problem 1

Exercise01 – Problem 2

```
95  /**
96   * @brief   BST에서 target_key값을 가진 노드를 찾아서 포인터를 반환한다.
97   *          만약, 그런 노드가 존재하지 않는다면 에러 메시지를 출력한 후 NULL을 반환한다.
98   *
99   * @param tree          검색을 수행 할 트리의 포인터
100  * @param target_key    검색 할 key값
101  * @return node_ptr     해당 key값을 가지는 노드의 포인터
102  */
103 node_ptr find(tree_ptr tree, element target_key){
104     node_ptr parent = NULL;
105     node_ptr candidate = tree->root;
106
107
108
109
110
111
112
113     return candidate;
114 }
```

Problem 2

Exercise01 – Problem 3, 4

```
116  /**
117   * @brief   BST에서 target_key값을 가진 노드를 찾아서 삭제 후 할당 해제한다.
118   *          만약, 그런 노드가 존재하지 않는다면 아무 행동도 하지 않는다.
119   *
120   * @param tree          검색을 수행 할 트리의 포인터
121   * @param target_key    검색 할 key값
122   * @return node_ptr     해당 key값을 가지는 노드의 포인터
123   */
124  void delete_node(tree_ptr tree, element target_key){
125      node_ptr parent;
126      node_ptr candidate;
127
128      // 해당 key값을 가진 노드가 존재하는지 검사해 본다.
129      candidate = find(tree, target_key);
130
131      if (candidate == NULL){
132          // 해당 key값을 가진 노드가 존재하지 않는 경우
133          return;
134      }
135
136      // 해당 key값을 가진 노드 candidate가 존재하는 경우
137      parent = candidate->parent;
138
139      if (candidate->left == NULL || candidate->right == NULL){
140          /*************************************
141          CASE 1 : candidate가 하나의 자식을 가지거나 자식을 가지지 않는 경우.
142          *************************************/
143
144
145
146
147          return;
148      }
149      else{
150          /*****************************************
151          CASE 2 : candidate가 정확히 두 개의 자식 노드를 가지는 경우
152          *****************************************/
153
154          node_ptr predecessor = candidate->left;
155          node_ptr parent_of_predecessor = candidate;
156
157
158
159
160          return;
161      }
162  }
```

Problem 3

Problem 4

<HINT 1>
자식이 하나 인 경우, 삭제 될 노드의 자리를 기존의 자식 노드로 대체하면 된다

<HINT 2>
두 개의 자식을 가진다
⇔ Predecessor는 항상 candidate의 자손이다.

Exercise01 – Example

테스트를 위한 코드는 아래 URL에서 Copy & Paste 하세요.

- https://gist.github.com/waps12b/93ec2a8d83cccc83cb5501ea0c730786

test_case_1() 함수의 실행 예시

- 보고서에는 test_case_2()함수를 실행 한 결과를 캡처해서 첨부하세요!