Lab 5

Data Structure

Lab 5 (due on Lab Session)

1. Do p5_1.c

2. No homework

Evaluation criteria

Category	Evaluation	
p5_1	100	
Total	100	

- *Use GCC 4.8 version or GCC 5.4 version.*
- *No score will be given if the gcc version is different.*

Lab 5

- You should finish p5_1 (InsertNode, PrintInorder) during the lab session and submit it to git before you leave.
- Folder name : Lab5
- code name: p5_1
- -15 score , if the folder, code names are wrong.
- -5 per code, if it does not use FILE I/O.
- Each code will be tested by 5 different input files.
- 20 score for each input, if you don’t get the answer you get 0 score.

Lab 5

Tree * insertNode(Tree *root, int key) Insert a new node with the key value into the tree. If the key already exists in the tree, print an error message.

void printInorder(Tree *root) Print the tree by inorder traversal.

Lab 5 - BST ADT

- **i x** insert a new key “x” into the binary search tree without duplication. If x already exists in the tree, print an error message.
- **pi** print the tree by inorder traversal.

Lab 5 – BST ADT

- Structure

```
typedef struct Tree{
        int value;
        struct Tree *left;
        struct Tree *right;
}Tree;
```

Lab 5. BST ADT – InsertNode, PrintInorder

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

typedef struct Tree{
    int value;
    struct Tree* left;
    struct Tree* right;
}Tree;

Tree* insertNode(Tree *root, int key);
void printInorder(Tree* root);
void deleteTree(Tree* root);

void main(int argc, char* argv[])
{
    FILE *fi = fopen(argv[1], "r");
    char cv;
    int key;

    Tree* root = NULL;

    while (!feof(fi))
    {
        fscanf(fi, "%c", &cv);
        switch(cv){
        case 'i':
            fscanf(fi, "%d", &key);
            root = insertNode(root, key);
            break;
```

```c
        case 'p':
            fscanf(fi, "%c", &cv);
            if(cv == 'i')
                printInorder(root);
            printf("\n");
            break;
        }
    }
    deleteTree(root);
}

void deleteTree(Tree* root)
{
    if(root == NULL)
        return;
    deleteTree(root->left);
    deleteTree(root->right);
    free(root);
}

Tree* insertNode(Tree* root, int key)
{


}

void printInorder(Tree *root)
{


}
```

Lab 5. Binary Search Tree ADT – Example1

- input file : lab5_input1.txt

```
i 7
i 1
i 9
i 10
i 1
pi
```

- Result

```
ds-04@ds04-VirtualBox:~/Downloads/week5$ gcc -o p5 p5_1.c
ds-04@ds04-VirtualBox:~/Downloads/week5$ ./p5 lab5_input1.txt
insert 7
insert 1
insert 9
insert 10
Insertion Error : There is already 1 in the tree.
1 7 9 10
```

Lab 5. Binary Search Tree ADT - InsertNode, PrintInorder

- program name : p5_1.c
- input : a list of numbers in a file.
- output : the corresponding result in the standard output.