Lab 6

Data Structure

Lab 6 (due on Lab Session)

1. Do p6_1.c

HW 6 (due on the day before the next Lab Session)

1. Do p6_2.c

Evaluation criteria

Category	Evaluation	
p6_1	50	
p6_2	50	
Total	100	

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

Lab6 – AVL Tree

- You should finish p6_1 (Insert, printInorder, DeleteTree) during the lab session and submit it to git before you leave.
- For p6_2 (SingleRotateWithLeft, SingleRotateWithRight, DoubleRotateWithLeft, DoubleRotateWithRight) you can submit it to the git later.
- Folder name : Lab6
- code name: p6_1, p6_2
- -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.
- 10 score for each input, if you don't get the answer you get 0 score.

Lab6 - AVL Tree

AVLTree Insert(ElementType X, AVLTree T) Insert a new node to the AVL Tree. If the key already exists in the tree, print an error message.

void printInorder(AVLTree T) Print the tree by inorder traversal. Print height of the node inside bracket.

void DeleteTree(AVLTree T) Free tree

Position SingleRotateWithLeft(Position node)

Position SingleRotateWithRight(Position node)

Position DoubleRotateWithLeft(Position node)

Position DoubleRotateWithRight(Position node)

Lab6 – AVL Tree

- Structure

```
struct AVLNode
{
        ElementType Element;
        AVLTree Left;
        AVLTree Right;
        int Height;
}
```

Lab6. AVL Tree – Main

```c
int main(int argc, char **argv){

        FILE *fp = fopen(argv[1], "r");
        AVLTree myTree = NULL;
        int num;

        if (fp == NULL){
                printf("There is no file : %s\n", argv[1]);
                exit(-1);
        }

        while (fscanf(fp, "%d", &num) != EOF){

                myTree = Insert(num, myTree);
                PrintInorder(myTree);
                printf("\n");
        }

        DeleteTree(myTree);
        return 0;
}
```

Lab6. AVL Tree - Insert, PrintInorder

- program name : p6_1.c
- input : a list of numbers in a file.
- output : the corresponding result in the standard output.

Lab6. AVL Tree – Example1

- input file : Lab6_input1.txt

```
7 5 3 10 23 4 20 21 22 23 24 25
```

- Result

```
7(0)
5(0) 7(1)
3(0) 5(1) 7(2)
3(0) 5(1) 7(2) 10(0)
3(0) 5(1) 7(2) 10(1) 23(0)
3(1) 4(0) 5(2) 7(3) 10(1) 23(0)
3(1) 4(0) 5(2) 7(3) 10(2) 20(0) 23(1)
3(1) 4(0) 5(2) 7(4) 10(3) 20(1) 21(0) 23(2)
3(1) 4(0) 5(2) 7(5) 10(4) 20(2) 21(1) 22(0) 23(3)
[Error] 23 already in the tree!
3(1) 4(0) 5(2) 7(5) 10(4) 20(2) 21(1) 22(0) 23(3)
3(1) 4(0) 5(2) 7(5) 10(4) 20(2) 21(1) 22(0) 23(3) 24(0)
3(1) 4(0) 5(2) 7(5) 10(4) 20(2) 21(1) 22(0) 23(3) 24(1) 25(0)
```

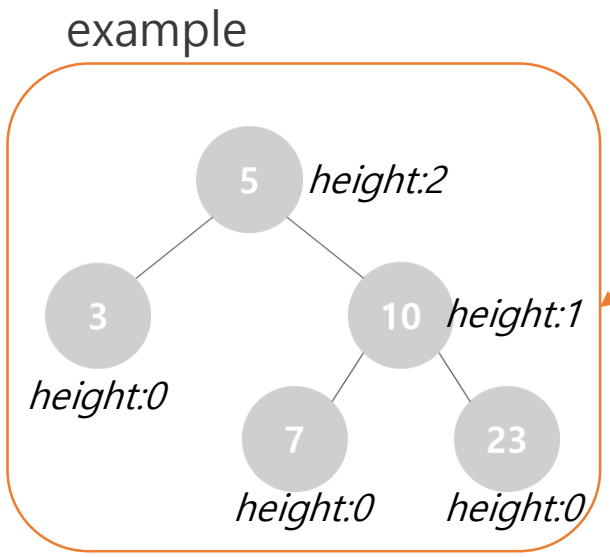
Lab6 AVL Tree – SingleLotateWithLeft, SingleLotateWithRight, DoubleLotateWithLeft, DoubleLotateWithRight

- program name : p6_2.c
- input : a list of numbers in a file.
- output : the corresponding result in the standard output.

Lab6. AVL Tree – Example2

- input file : Lab6_input2.txt

```
7 5 3 10 23 4 20 21 22 23 24 25
```



- Result

```
7(0)
5(0) 7(1)
3(0) 5(1) 7(0)
3(0) 5(2) 7(1) 10(0)
3(0) 5(2) 7(0) 10(1) 23(0)
3(1) 4(0) 5(2) 7(0) 10(1) 23(0)
3(1) 4(0) 5(3) 7(0) 10(2) 20(0) 23(1)
3(1) 4(0) 5(3) 7(0) 10(2) 20(0) 21(1) 23(0)
3(1) 4(0) 5(3) 7(0) 10(1) 20(0) 21(2) 22(0) 23(1)
[Error] 23 already in the tree!
3(1) 4(0) 5(3) 7(0) 10(1) 20(0) 21(2) 22(0) 23(1)
3(1) 4(0) 5(3) 7(0) 10(1) 20(0) 21(2) 22(0) 23(1) 24(0)
3(1) 4(0) 5(2) 7(0) 10(1) 20(0) 21(3) 22(0) 23(2) 24(1) 25(0)
```