Data Structure HW8

start date = 10월 30일 (금)

due date = 11월 13일 (금) 23시 59분 59초

submit 방법 : submit pem_ta hw3a or hw3b (1분반 : a, 2분반 : b)

※ 제출 파일 : hw8.cpp, bst.h, 학번.tex, 학번.pdf, image files

※ 제출 시 실행파일을 지우고 제출합니다.

※ 지정한 파일명과 다를 경우 0점 처리합니다.

1. dsdir8 이라는 디렉토리에서 다음과 같은 makefile을 만든다.

`cat makefile`

```
hw8: hw8.o
  g++ -o hw8 hw8.o
hw8.o: bst.h
```

2. hw8.cpp 는 다음과 같다.

`cat hw8.cpp`

```
#include "bst.h"

using namespace std;

int main(void){
  BST<string, double> tree, tree1;
  string command, str;
  double dval;

  while(cin >> command){
    if(command == "insert"){
      cin >> str >> dval;
      tree.Insert(str, dval);
    }
    else if(command == "delete"){
      cin >> str;
      tree.Delete(str);
    }
    else if(command == "print"){
      cout << "Inorder traversal : ";
      tree.Inorder();
      cout << endl;
    }
    else if(command == "find"){
      cin >> str;
      if(tree.Find(str, dval)){
        cout << "The value of " << str << " is " << dval << endl;
      }
      else{
        cout << "No such key : " << str << endl;
      }
    }
    else if(command == "max"){
      tree.Max();
    }
    else if(command == "min"){
      tree.Min();
    }
    else cout << "Invalid command : " << command << endl;
  }
}
```

3. 테스트 케이스를 생성한다.

`cat bst.in`

```
insert haha 23
insert huhu 77
insert hihi 12
insert hoho 45
insert ha 12
insert haha 99
print
find haha
find huhu
find hoho
find hohoho
max
min
```

`cat bst.in2`

```
insert haha 23
insert huhu 77
insert hihi 12
insert hoho 45
insert ha 12
insert haha 99
print
max
delete hihi
delete huhu
print
max
find haha
find huhu
find hoho
find hohoho
```

4. 다음과 같이 동작하는 프로그램을 작성하려고 한다.

`make hw8`

`hw8 < bst.in`

```
Inorder traversal : ha:12 haha:99 hihi:12 hoho:45 huhu:77
The value of haha is 99
The value of huhu is 77
The value of hoho is 45
No such key : hohoho
Maximum value is huhu:77
Minimun value is ha:12
```

`hw8 < bst.in2`

```
Inorder traversal : ha:12 haha:99 hihi:12 hoho:45 huhu:77
Maximum value is huhu:77
Inorder traversal : ha:12 haha:99 hoho:45
Maximum value is hoho:45
The value of haha is 99
No such key : huhu
The value of hoho is 45
No such key : hohoho
```

5. 아래 bst.h 파일을 완성하시오.

`cat bst.h`

```
#include <iostream>
#include <string>

using namespace std;

template <class K, class E>
struct Node{
  Node(K ky, E el, Node<K, E> *left = 0, Node<K, E> *right = 0) : key(ky), element(el), leftChild(left), rightChild(right) {}
  Node<K, E> *leftChild;
```

```
  K key;
  E element;
  Node<K, E> *rightChild;
};

template <class K, class E>
class BST{
public:
  BST(){ root = 0; }
  void Insert(K &newkey, E &el){ Insert(root, newkey, el); }
  void Inorder(){ Inorder(root); }
  void Delete(K &oldkey){ Delete(root, root, oldkey); }
  bool Find(const K &, E &);
private:
  void Visit(Node<K, E> *);
  void Insert(Node<K, E> *& , K & , E &); // 구현
  void Inorder(Node<K, E> *);
  Node<K, E> *Delete(Node<K, E> *&, Node<K, E> *&, K &); // 구현
  Node<K, E> *Max(Node<K, E>*& ); // 구현
  Node<K, E> *Min(Node<K, E>*& ); // 구현
  Node<K, E> *root;
};

template <class K, class E>
void BST<K, E>::Visit(Node<K, E> *ptr){
  cout << ptr -> key << ':' << ptr -> element << ' ';
}

template <class K, class E>
void BST<K, E>::Inorder(Node<K, E> *currentNode){
  if(currentNode){
    Inorder(currentNode -> leftChild);
    Visit(currentNode);
    Inorder(currentNode -> rightChild);
  }
}

template <class K, class E>
bool BST<K, E>::Find(const K &k, E &e){
  Node<K, E> *temp = root;

  while(temp){
    if(k > temp -> key){
      temp = temp -> rightChild;
    }
    else if(k < temp -> key){
      temp = temp -> leftChild;
    }
    else{
      e = temp -> element;
      return true;
    }
  }

  return false;
}
```

※ Insert, Delete, Min, Max 함수를 Threaded binary tree 방식으로 구현하시오.

6. 주의사항

- 이전 과제에서 작성한 code 사용 가능.
- 문제에 명시된 함수 이외에 추가적인 함수 생성 불가.
- 제공된 Node와 함수를 수정할 수 있다.
- 제공된 테스트 케이스와 채점에 사용되는 테스트 케이스는 다르다.
- Threaded binary tree inorder 를 제공하지 않지만, 해당 방식으로 순회가 되지 않으면 0점 처리.

7. 질문

- pemta806@gmail.com 으로 질문한다.
- 질문은 최대한 자세하게 한다.
- 코드 첨부가 필요하다면 이미지가 아닌 텍스트로 첨부한다.