레포트#3

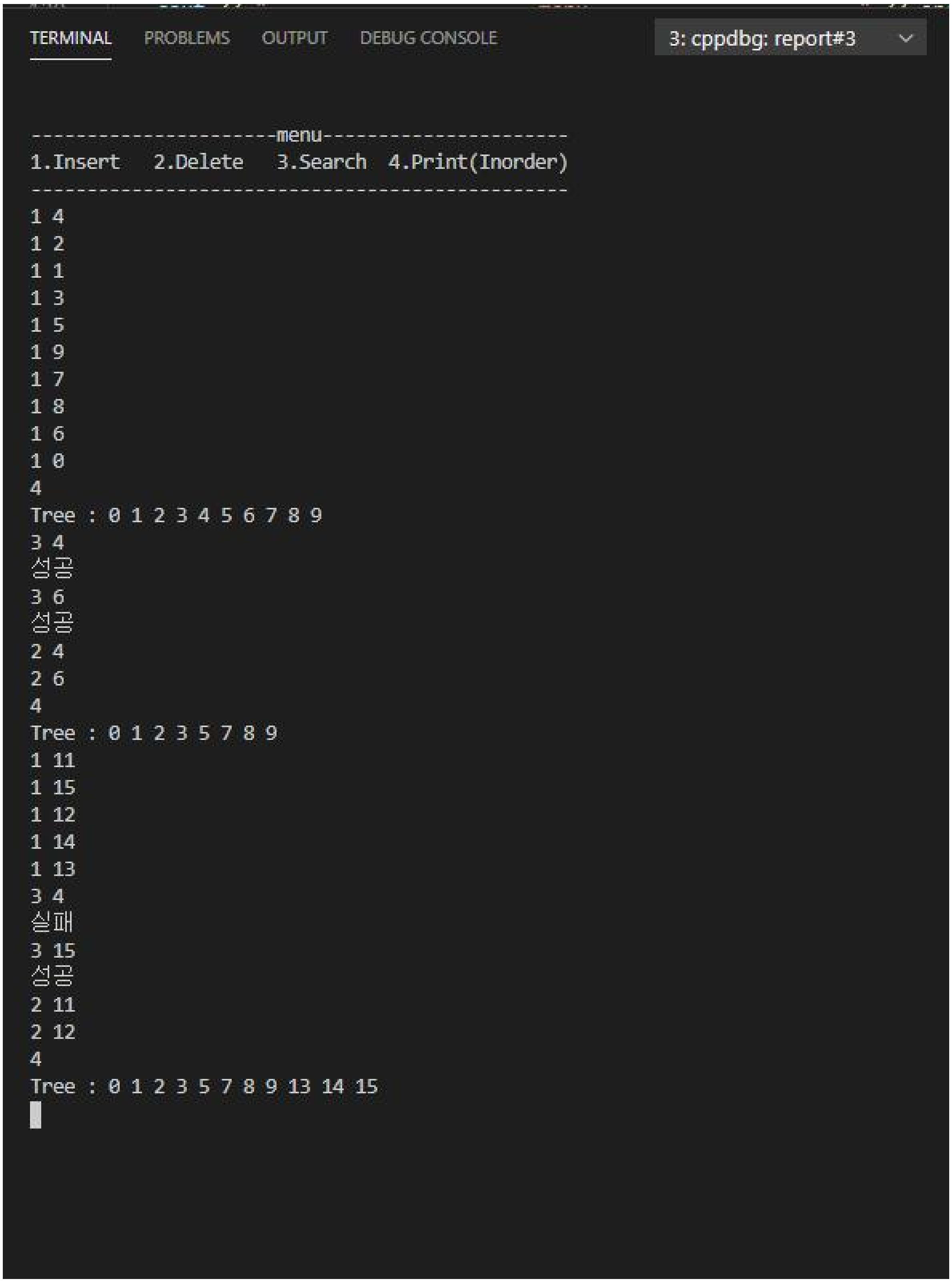
자료구조(SW) 2분반
소프트웨어학과
32170578 김산

<코드>

```
#include<iostream>
#include<stdbool.h>
using namespace std;
template<class T>class BST;
template<class T>class BstNode{
friend class BST<T>;
private:
	BstNode *LeftChild;
	T data;
	BstNode *RightChild;
public:
	BstNode(T value =0, BstNode* left =0,BstNode* right =0){
		this->data = value;
		this->LeftChild = left;
		this->RightChild = right;
	}
};
template<class Type>class BST{
private:
	BstNode<Type>*root;
public:
	BST(): root(0){};
	bool Insert(const Type& x); //삽입함수
	bool IterSearch(const Type& x); //탐색함수
	void Delete(const Type& x);//삭제함수
	BstNode<Type>* Delete(BstNode<Type>*p,const Type& x);
	BstNode<Type>* FindMinNode(BstNode<Type>* p); //오른쪽 서브트리에서 가장작은 노드
	void InorderPrint(); //중위순회
	void InorderPrint(BstNode<Type>*p);
};
/*삽입 함수*/
template<class Type>
bool BST<Type>::Insert(const Type& x){
	BstNode<Type>*p = root;
	BstNode<Type>*q =0;
	while(p) {
		q = p;
		if (x == p->data) return false; // x가 이미 존재
		if (x < p->data) p = p->LeftChild;
		else p = p->RightChild;
	}
	p =new BstNode<Type>; // 삽입을 수행
	p->LeftChild = p->RightChild =0; p->data = x;
	if (!root) root = p;
	else if (x < q->data) q->LeftChild = p;
	else q->RightChild = p;
	return true;
}
/*탐색함수*/
template<class Type>
bool BST<Type>::IterSearch(const Type& x){
	for(BstNode<Type>*t=root; t;){
		if(x == t->data) return true;
		if(x < t->data) t = t->LeftChild;
		else t = t->RightChild;
	}
	return false;
}
/*삭제함수*/
template<class Type>
void BST<Type>::Delete(const Type& x){
	Delete(root,x);
}
template<class Type>
BstNode<Type>* BST<Type>::Delete(BstNode<Type>*p,const Type& x){
	BstNode<Type>* tmp =NULL;
	if(p ==NULL) return NULL;
	if(p->data > x) p -> LeftChild = Delete(p->LeftChild,x);
	else if(p->data < x){
		p->RightChild = Delete(p->RightChild,x);
	}
	else{
		if(p->RightChild !=NULL && p->LeftChild !=NULL){ //두개의 자식을 가질경우
			tmp = FindMinNode(p->RightChild);
			p->data = tmp->data; //오른쪽서브트리에서 가장 작은노드로 대체
			p->RightChild = Delete(p->RightChild,tmp->data);
		}
		else{
			tmp = (p->LeftChild==NULL) ? p->RightChild : p->LeftChild;
			free(p);
			return tmp;
		}
	}
}
template<class Type>//오른쪽 노드에서 가장 작은 노드 찾기
BstNode<Type>* BST<Type>::FindMinNode(BstNode<Type>* p){
	BstNode<Type>* tmp = p;
	while(tmp->LeftChild !=NULL) tmp = tmp->LeftChild;
	return tmp;
}
/*중위순회*/
template<class Type>
void BST<Type>::InorderPrint(){
	InorderPrint(root);
}
template<class Type>
void BST<Type>::InorderPrint(BstNode<Type>*p){
	if(p){
		InorderPrint(p->LeftChild);
		cout << p->data <<" ";
		InorderPrint(p->RightChild);
	}
}
int main(){
	BST<int> t;
	int menu;
	int value;
	cout <<endl;
	cout <<"----------------------menu----------------------"<<endl;
```

```
        cout <<"1.Insert     2.Delete     3.Search    4.Print(Inorder)"<<endl;
        cout <<"-----------------------------------------------"<<endl;
        while(1){
                cin >> menu;
                if(menu ==1 || menu ==2 || menu ==3) cin >> value;
                switch (menu)
                {
                case 1:
                        t.Insert(value);
                        break;
                case 2:
                        t.Delete(value);
                        break;
                case 3:
                        if(t.IterSearch(value)) cout<<"성공"<<endl;
                        else cout <<"실패"<<endl;
                        break;
                case 4:
                        cout <<"Tree : ";
                        t.InorderPrint();
                        cout <<endl;
                        break;
                default:
                        cout <<"wrong input, exit program"<<endl;
                        return 0;
                }
        }
}
```

<실행 화면>



```
----------------------menu---------------------
1.Insert   2.Delete   3.Search  4.Print(Inorder)
-----------------------------------------------
1 4
1 2
1 1
1 3
1 5
1 9
1 7
1 8
1 6
1 0
4
Tree : 0 1 2 3 4 5 6 7 8 9
3 4
성공
3 6
성공
2 4
2 6
4
Tree : 0 1 2 3 5 7 8 9
1 11
1 15
1 12
1 14
1 13
3 4
실패
3 15
성공
2 11
2 12
4
Tree : 0 1 2 3 5 7 8 9 13 14 15
```