1. Exercise 6

다음과 같은 규격 명세를 사용하여 2개의 정렬 리스트 ADT를 병합시키는 클라이언트 함수를 작성하여라.

클라이언트 함수 : 클래스의 멤버함수가 아님, 클래스 멤버함수들을 사용하는 외부 전역함수

MergeList(SortedType list1, SortedType list2, SortedType& result)

함수: 2개의 정렬 리스트를 Merge해서 세 번째 정렬 리스트를 만든다.

조건: list1과 list2는 초기화되어 있고 ComparedTo라는 함수를 사용해서 키에 의해 정 렬되어 있다. list1과 list2는 같은 키를 갖지 않는다.

결과: 결과는 list1과 list2의 모든 요소를 가진 정렬 리스트이다.

a. 함수를 작성하여라

```
int MergeList(SortedType list1, SortedType list2, SortedType& result)
{
    list1.ResetList();
    list2.ResetList();
    if (result.IsFull())
        return -1;
    for (int i = 0; i < list1.LengthIs(); i++)
    {
        ItemType temp;
        list1.GetNextItem(temp);
        result.InsertItem(temp);
    }

    for (int i = 0; i < list2.LengthIs(); i++)
    {
        ItemType temp;
        list2.GetNextItem(temp);
        result.InsertItem(temp);
    }
```

```
int main()
{
    SortedType sorted_item_a{};
    SortedType sorted_item_b{};
    SortedType sorted_merged{};
    sorted_item_a.ResetList();
    sorted_item_b.ResetList();
    sorted_merged.ResetList();

    ItemType item{};
    for (int i = 0; i < 5; i++)
    {
        item.Initialize(i);
        sorted_item_a.InsertItem(item);
    }
    for (int i = 5; i < 10; i++)
    {
        item.Initialize(i);
        sorted_item_b.InsertItem(item);
    }
    MergeList(sorted_item_b, sorted_item_a, sorted_merged);
    for (int i = 0; i < sorted_merged.LengthIs(); i++)
    {
        ItemType temp{};
        sorted_merged.GetNextItem(temp);
        temp.Print(cout);
    }

    // interupt
    int a;
    cin >> a;
```

b. Big-O 표기법으로 알고리즘을 표현하여라.

$O(N^2)$이다.

list1에서 0부터 length까지 반복하니 N번, for문.

inset Item에서 index를 찾는 과정에서 n번, array에 삽입하는 과정에서 n번 2n

+

List2에서 0부터 length까지 반복하니 N번, for문.

inset Item에서 index를 찾는 과정에서 n번, array에 삽입하는 과정에서 n번 2n

worst case 총계산 량은 $2N^2 + 2N^2$ => *약* $4N^2$ *번이다.*

2. Binary Search

A. 이진 탐색(binary search)를 위한 함수 BinarySearch()를 구현한다.

```
//return index
int BinarySearch(int array[], int length, int value)
{
    int first_index = 0;
    int last_index = length - 1;
    while (first_index <= last_index)
    {
        int middle_index = (first_index + last_index) / 2;

        if (array[middle_index] == value)
            return middle_index;
        else if (array[middle_index] > value)
            last_index = middle_index - 1;
        else if (array[middle_index] < value)
            first_index = middle_index + 1;
    }
    return -1;
}
```

B. BinarySearch를 '수정'하여 찾고자 하는 값보다 작거나 같은 값들 중에서 가장 큰 값을 리턴하게 하려면 어떻게 하는가?

return -1; 부분을 수정

```
//return equal or smaller value in the array
int BinarySearch(int array[], int length, int value)
{
    int first_index = 0;
    int last_index = length - 1;
    while (first_index <= last_index)
    {
        int middle_index = (first_index + last_index) / 2;

        if (array[middle_index] == value)
            return middle_index;
        else if (array[middle_index] > value)
            last_index = middle_index - 1;
        else if (array[middle_index] < value)
            first_index = middle_index + 1;
    }
    if (array[last_index] < value && value < array[last_index+1])
        return array[last_index];
    else
        return array[last_index + 1];
}
```

C. BinarySearch를 '수정'하여 찾고자 하는 값보다 크거나 같은 값들 중에서 가장 작은 값을 리턴하게 하려 면 어떻게 하는가?

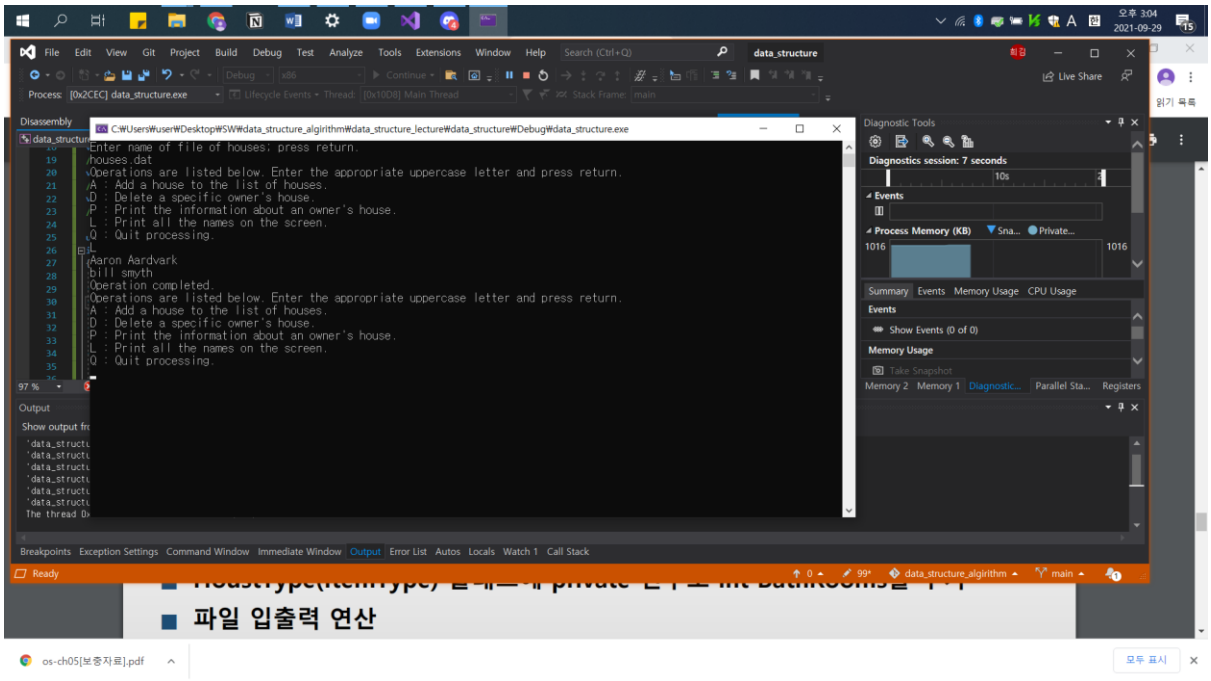
return -1; 부분을 수정

```
//return equal or smaller value in the array
int BinarySearch(int array[], int length, int value)
{
    int first_index = 0;
    int last_index = length - 1;
    while (first_index <= last_index)
    {
        int middle_index = (first_index + last_index) / 2;

        if (array[middle_index] == value)
            return middle_index;
        else if (array[middle_index] > value)
            last_index = middle_index - 1;
        else if (array[middle_index] < value)
            first_index = middle_index + 1;
    }
    if (array[first_index-1] < value && value < array[first_index])
        return array[first_index];
    else
        return array[first_index - 1];
}
```

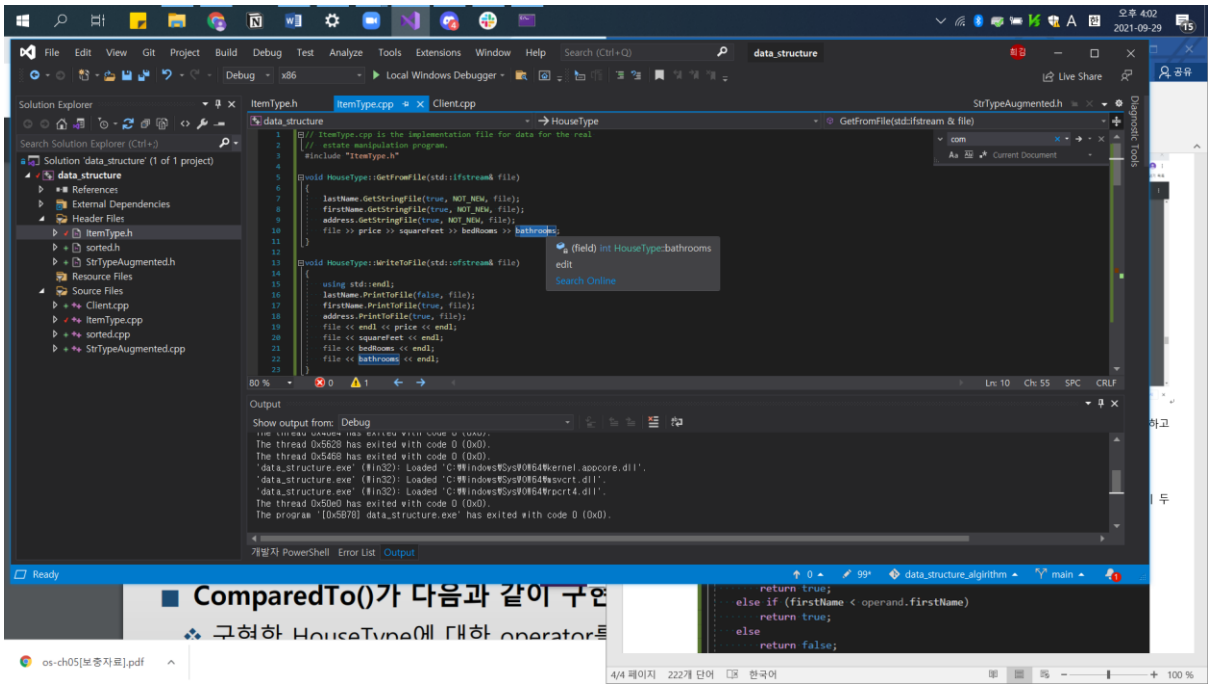
3. Case study

교재 Case Study에 있는 프로그램 소스를 자세히 읽고 프로그램을 실제로 실 행해 본다.



A. HouseType에 bathroom의 개수를 나타내는 bathrooms 변수를 추가하고 이를 입력하고 출력할 수 있도록 프로그램을 수정한다.

자세한 사항은 ItemType.cpp



B. HouseType에 relational operator < 와 ==를 overloading하고 ComparedTo 함수를 이 두

연산자를 이용하여 구현하도록 수정한다.

```
bool HouseType::operator < (const HouseType& operand)const
{
    if (lastName < operand.lastName)
        return true;
    else if (firstName < operand.firstName)
        return true;
    else
        return false;
}

bool HouseType::operator == (const HouseType& operand)const
{
    if (lastName == operand.lastName && firstName == operand.lastName)
        return true;
    else
        return false;
}
```