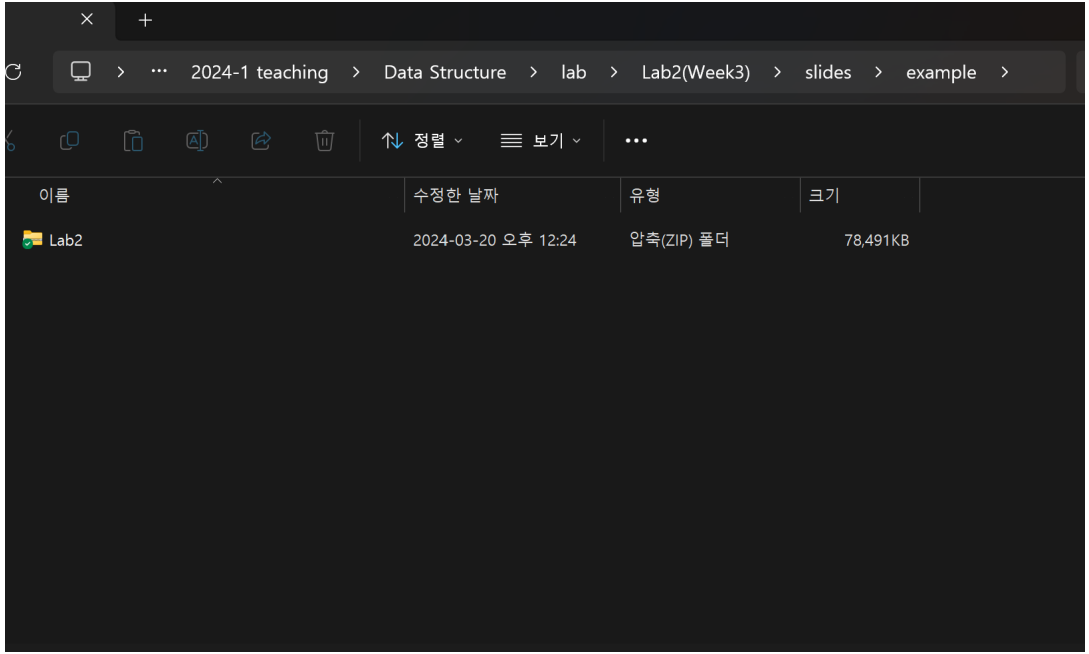
Data Structure Programming Exercise

Lab #2
Class on 3/20
Assignment by 3/26 (23:59)

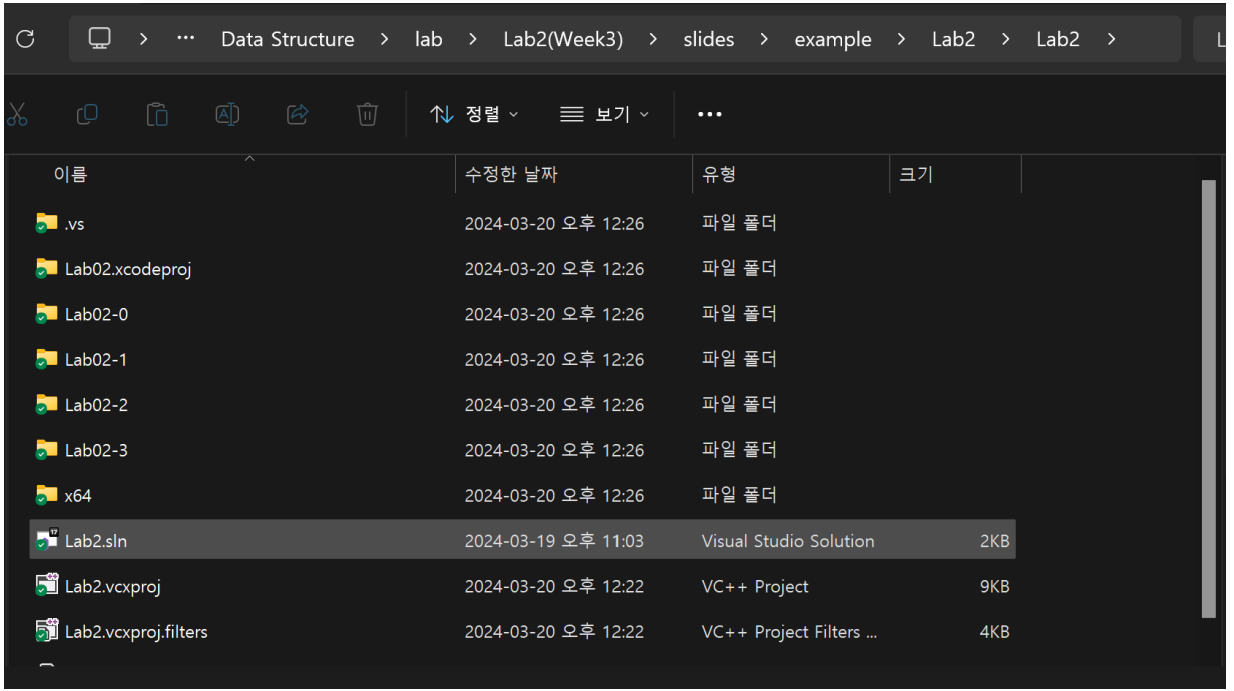
Exercise #0

- Problems
  - Follow the steps “how to do assignments”
  - Step1: Download the template



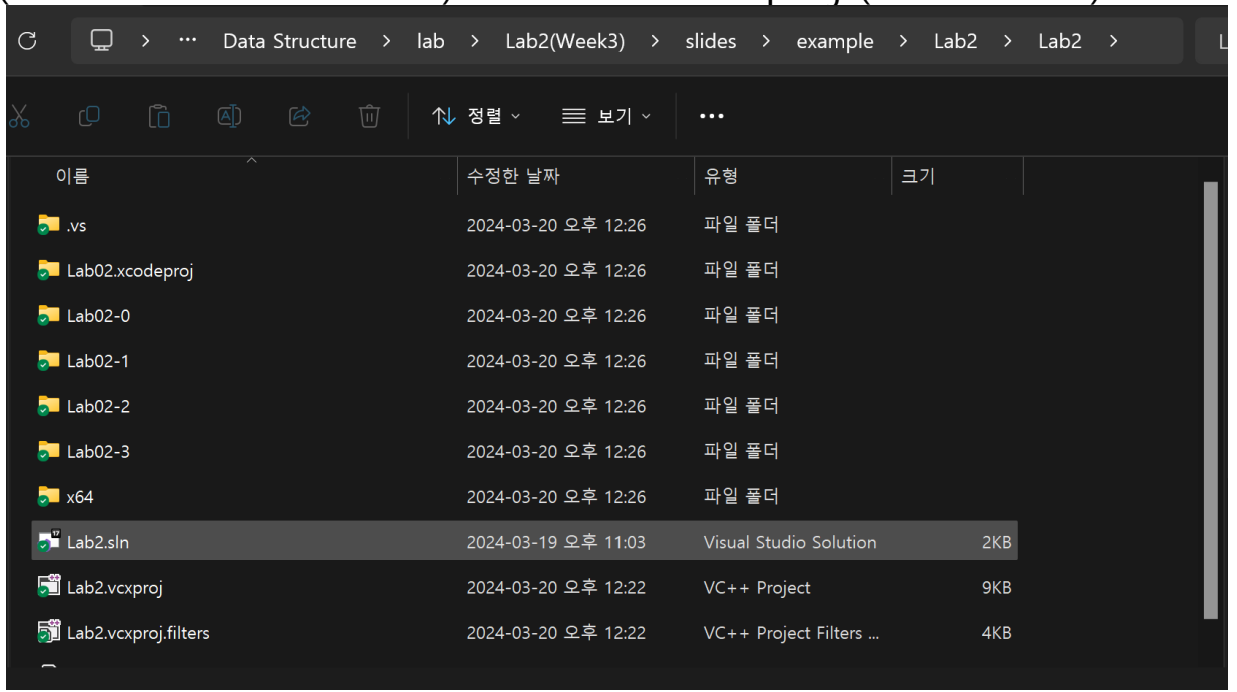
Exercise #0

- Problems
  - Step2: Unzip the file



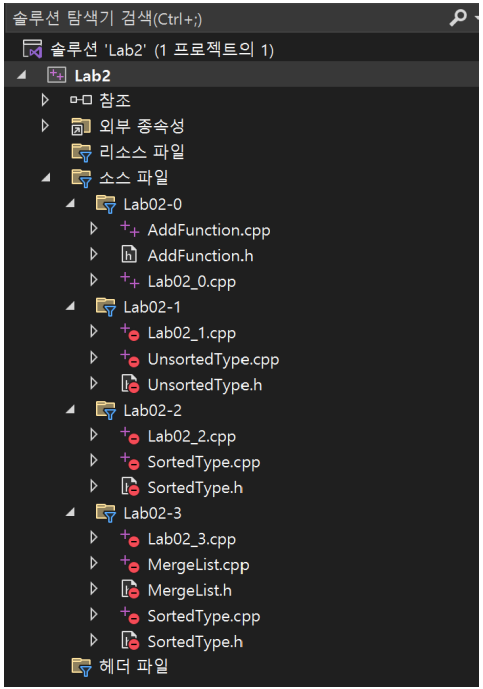
Exercise #0

- Problems
  - Step3: Open Lab2.sln (Windows-VisualStudio) or Lab02.xcodeproj (Mac-Xcode)



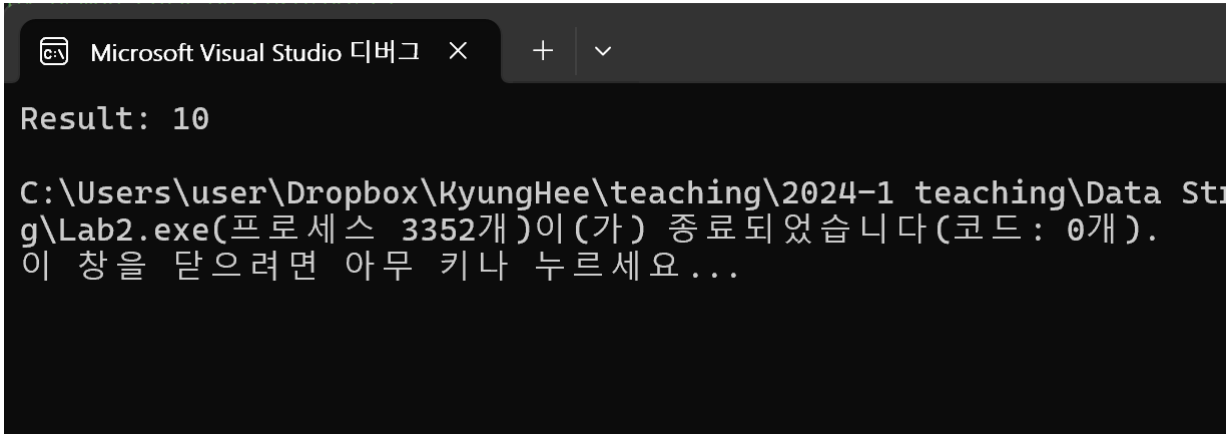
Exercise #0

- Problems
  - Step4: Excludes all files (03-1, 03-2, 03-3) except for Lab03-0.cpp from Build.



Exercise #0

- Problems
  - Step5: Run it.



Exercise #0

- Problems
  - Step5-1: if you have “PDB API” issues, you need to set “symbol server” (ecampus notice)

Exercise #0

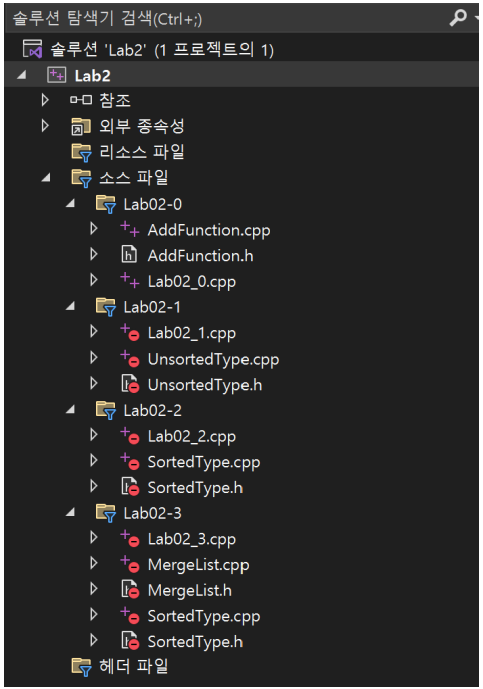
- Problems
  - Step5-2: if “PDB API” issue continues, try “[build] – [clean solution] (솔루션 정리)” and run it.

Exercise #0

- Problems
  - Step5-3: if you use older version of Visual Studio, you need to set the appropriate project version (ecampus notice)

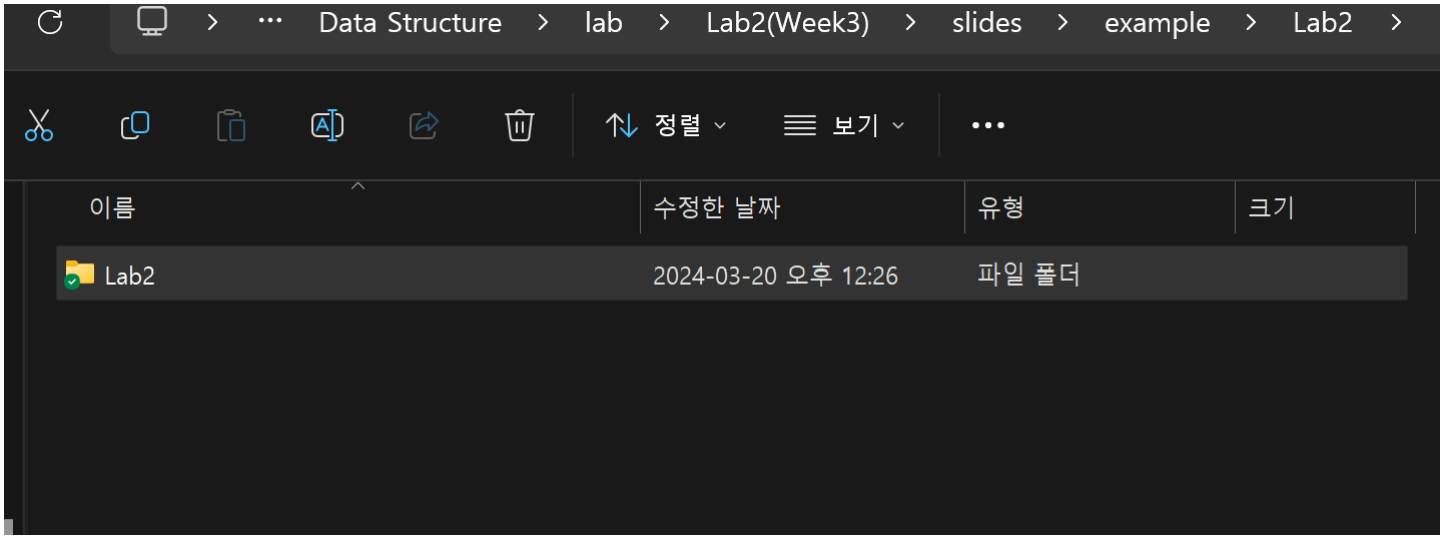
Exercise #0

- Problems
  - Step6: Do your best on each Lab question.



Exercise #0

- Problems
  - Step7: When you finish, move to the parent folder of .sln.



Exercise #0

- Problems
  - Step8: Zip “Lab2” folder again and rename it as “yourID_name.zip”

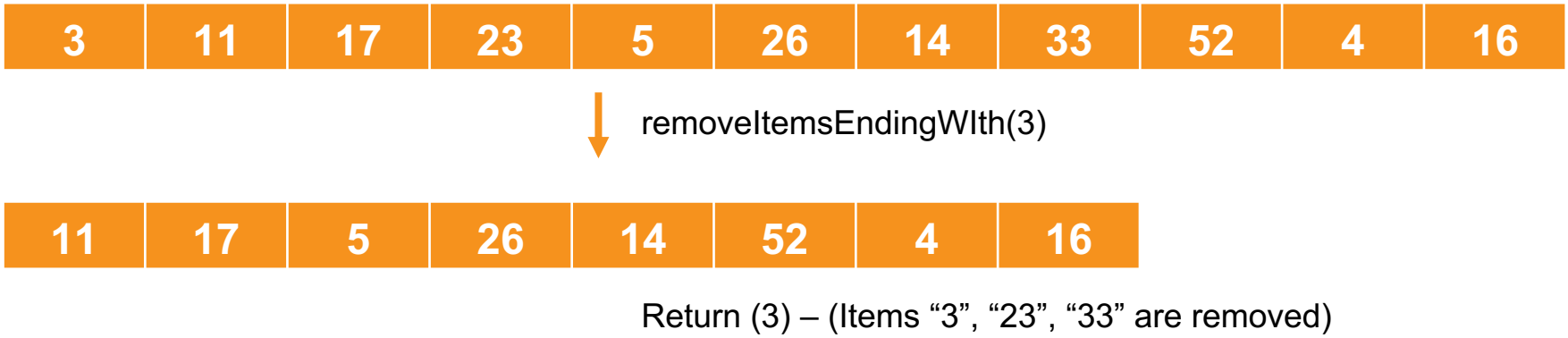
Exercise #0

- Problems
  - Step9: Submit the zip file and double check the submitted file.

Exercise #1

- Problems
  - Implement removeItemsEndingWith(int endDigit).

  - This function removes all items ending with "endDigit" and returns the number of elements removed.



Exercise #1

- How to submit (source codes)
  - Files: Lab02-1.cpp, UnsortedType.h, UnsortedType.cpp
  - Do: Implement removeItmesEndingWith( ) function in UnsortedType.cpp.

Exercise #2

- Problems
  - Implement (1) BinarySearch(int item) and (2) BinarySearchNearest(int item). Assume that there are **NO** duplicated elements in the list.

  - (1) BinarySearch(int item) is the function we already learned in the class.
    - The only difference is that "it returns the index of the item" if found
    - Otherwise, it returns -1

BinarySearch(5) → return 1 (the index of "5" is 1)
BinarySearch(20) → return 4 (the index of "20" is 4)
BinarySearch(4) → return -1 ("4" not exists in the list)
BinarySearch(15) → return -1 ("15" not exists in the list)

Exercise #2

- Problems
  - (2) BinarySearchNearest(int item) is similar to BinarySearch( )
    - It also returns "the index of the item" if found
    - Otherwise, it returns the index of the item nearest to "item"
    - If it has > two nearest elements, it can return any of them

BinarySearch(5) → return 1 (the index of "5" is 1)
BinarySearch(20) → return 4 (the index of "20" is 4)

BinarySearch(4) → return 1 ("5" is the nearest to "4" and index of "5" is 1)
BinarySearch(15) → return 3 ("12" is the nearest to "15" and index of "12" is 3)

BinarySearch(24) → return 5 or return 6
(Both "21" and "27" are the nearest to "24")
(Thus, it can return index of "21" (5) or index of "27" (6))

Exercise #2

- How to submit (source codes)
  - Files: Lab02-2.cpp, SortedType.h, SortedType.cpp
  - Do: Implement BinarySearch( ) and BinarySearchNearest( ) functions in SortedType.cpp.

Exercise #3

- Problems
  - Implement MergeList(SortedType list1, SortedType list2).

  - This function merges two input sorted lists, and returns the result

list1

3	11	17	23

list2

1	5	11	18	20

MergeList(list1, list2)

1	3	5	11	11	17	18	20	23

Exercise #3

- HINT:
  - MergeList( ) function is not a member function of "SortedType".
  - This means, in MergeList( ), you cannot directly access private members of "SortedType".
  - You can use getItem( ) instead.

```
// Wrong Example:
for (int i=0; I < list1.size( ); i++){
    list1.data[i] .....
}

// Correct Example:
for (int i=0; I < list1.size( ); i++){
    list1.getItem(i) .....
}
```

Exercise #3

- How to submit (source codes)
  - Files: Lab02-3.cpp, SortedType.h, SortedType.cpp, MergeList.h, MergeList.cpp
  - Do: Implement MergeList( ) function in MergeList.cpp.