EE2410 Data Structure Coding HW #6 – Sorting Hashing (Chapter 7 ~ 8)
due date 6/23/2024 (Sun.), 23:59

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program, i.e., write a client main() to demonstrate all functions you designed using example data.

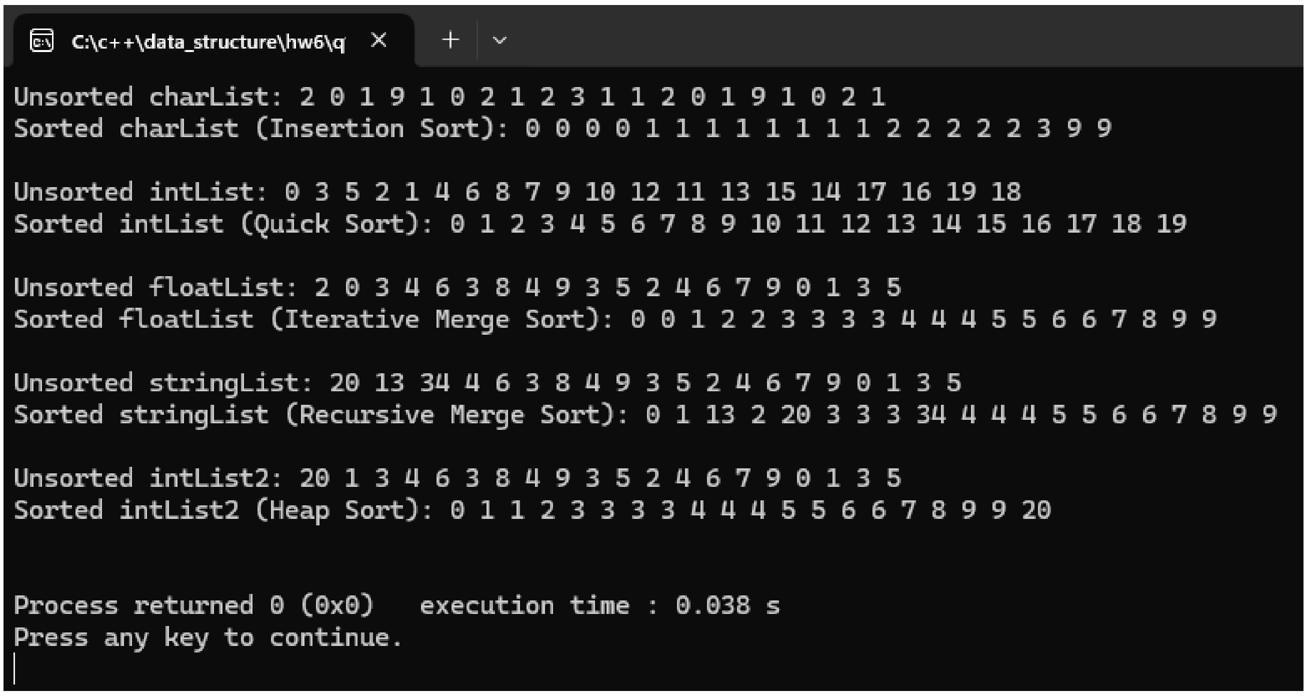
Sorting:

1. (50%)

Write a C++ program to perform 5 different sorting: insertion sort, quick sort, iterative merge sort, recursive merge sort, and heap sort, on lists of characters, integer, floating point numbers, and C++ strings.

1. You need to write the 5 sorting function templates (refer to example programs in textbook or pptx)
2. Randomly generate a list of 20 characters as an input unsorted list.
3. Randomly generate a list of 20 integers as an input unsorted list.
4. Randomly generate a list of 20 floating point numbers as an input unsorted list.
5. Randomly generate a list of 20 string objects as an input unsorted list.

Show your results using the above 4 lists in your program.



```
Unsorted charList: 2 0 1 9 1 0 2 1 2 3 1 1 2 0 1 9 1 0 2 1
Sorted charList (Insertion Sort): 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 3 9 9

Unsorted intList: 0 3 5 2 1 4 6 8 7 9 10 12 11 13 15 14 17 16 19 18
Sorted intList (Quick Sort): 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Unsorted floatList: 2 0 3 4 6 3 8 4 9 3 5 2 4 6 7 9 0 1 3 5
Sorted floatList (Iterative Merge Sort): 0 0 1 2 2 3 3 3 3 4 4 4 5 5 6 6 7 8 9 9

Unsorted stringList: 20 13 34 4 6 3 8 4 9 3 5 2 4 6 7 9 0 1 3 5
Sorted stringList (Recursive Merge Sort): 0 1 13 2 20 3 3 3 34 4 4 4 5 5 6 6 7 8 9 9

Unsorted intList2: 20 1 3 4 6 3 8 4 9 3 5 2 4 6 7 9 0 1 3 5
Sorted intList2 (Heap Sort): 0 1 1 2 3 3 3 3 4 4 4 5 5 6 6 7 8 9 9 20


Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

Hashing:

2. (50%)

Write a C++ program to implement **two simple symbol tables** (dictionaries) using hash table with linear probing for collision and hash table with chaining. For simplicity,

a. Consider storing only the key (need not consider the (key, value) pair) in the symbol tables.

b. Furthermore, the key is a **variable-length character array** where the first character of the key is an alphabet, e.g., abc, abcde, b, bye, cool,…

c. Consider a **simple hash function using only the first character of key to hash**, so h(abcde) = h(abc), h(b) = h(bye),.., etc. Therefore, collision can happen frequently.

d. The initial hash table size can be set to 26 since we have 26 alphabets which are the hashed keys.

Create 2 symbol table classes for linear probing and chaining, respectively. Both must implement at least the following functions:

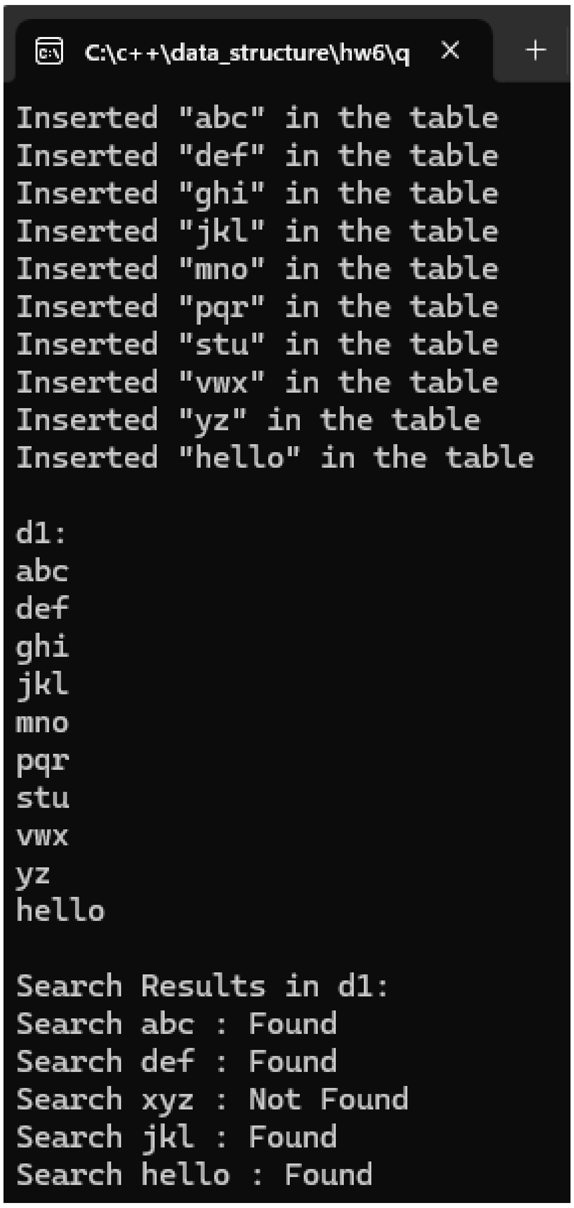
Constructor,
Insert(key)
Search(key)

You may add other functions needed in your program.

Your main function may contains code like:

SymbolTable1 d1;
Setup at least 10 key objects
Insert those 10 keys into d1.
Display d1
Demo the search function of d1 (try at least 5 keys)

SymbolTable2 d2;
Setup at least 10 key objects
Insert those 10 keys into d1.
Display d2
Demo the search function of d2(try at least 5 keys)

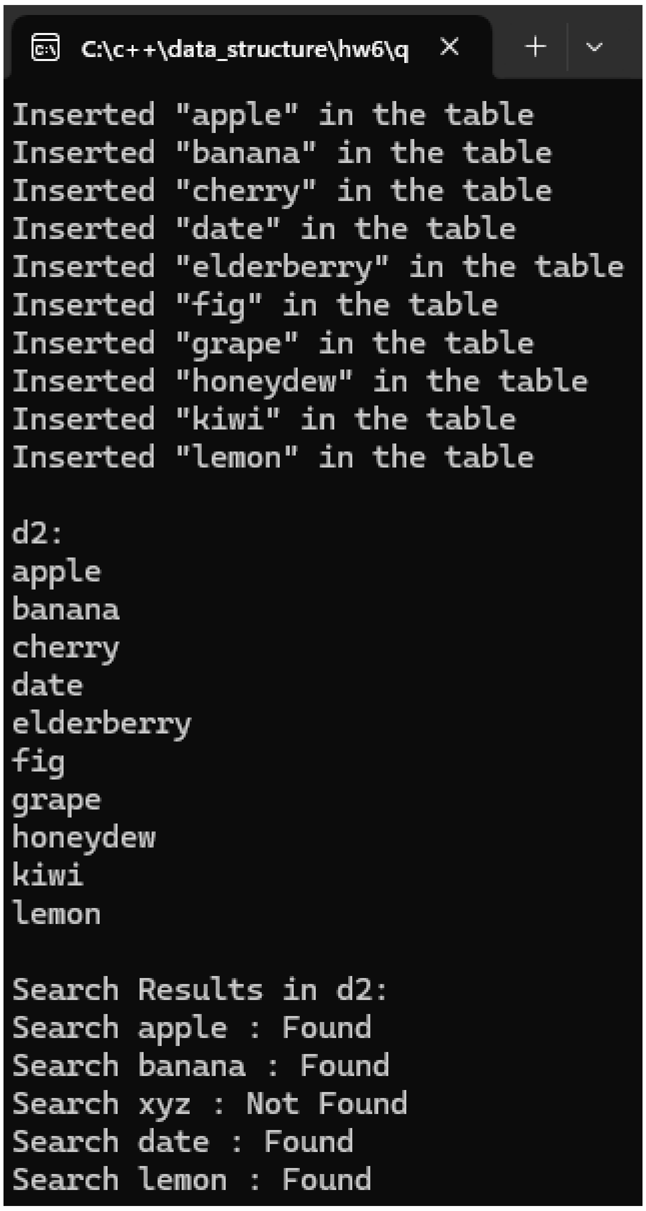


C:\c++\data_structure\hw6\q

```
Inserted "abc" in the table
Inserted "def" in the table
Inserted "ghi" in the table
Inserted "jkl" in the table
Inserted "mno" in the table
Inserted "pqr" in the table
Inserted "stu" in the table
Inserted "vwx" in the table
Inserted "yz" in the table
Inserted "hello" in the table

d1:
abc
def
ghi
jkl
mno
pqr
stu
vwx
yz
hello

Search Results in d1:
Search abc : Found
Search def : Found
Search xyz : Not Found
Search jkl : Found
Search hello : Found
```



C:\c++\data_structure\hw6\q

```
Inserted "apple" in the table
Inserted "banana" in the table
Inserted "cherry" in the table
Inserted "date" in the table
Inserted "elderberry" in the table
Inserted "fig" in the table
Inserted "grape" in the table
Inserted "honeydew" in the table
Inserted "kiwi" in the table
Inserted "lemon" in the table

d2:
apple
banana
cherry
date
elderberry
fig
grape
honeydew
kiwi
lemon

Search Results in d2:
Search apple : Found
Search banana : Found
Search xyz : Not Found
Search date : Found
Search lemon : Found
```