Hash-Map Word Count Project

cs 2420

Purpose

To gain experience designing, implementing, and using a Hash-Map.

Overview

You are to design the Hash-Map portion of a word-frequency count application. The app will open the AliceInWonderland.txt file (provided) (the code in main() assumes it to be in "default" location. For Visual Studio, this is where the source files are.) and parse the individual words from that file. Each word will be added to the Hash-Map with the GetKeyvalue() and SetKeyValue() member functions. The text word will be the "key". Initially, that key will not be in the map, so SetKeyValue () will add it to the map and set its initial count to 1. Subsequently, when the same key processed, GetkeyValue() will tell what the current count is and then a call to SetKeyValue() will increment the count by one.

Once the entire text of Alice In Wonderland has been added to the map, the 25 most frequently occurring words will be found and displayed. The FindLargestWordCount() function in Main.cpp will do this, but it will need to iterate over all keys in the map to find the one with the largest value. You will have to provide an **external iterator** to assist in this. The HashMap class has both a begin() and an end() function which return Iterators, similar to your external iterator project.

Your Part

You will have to write:

- Iterator.cpp
- HashMap.cpp

Note that the header information for Iterator.cpp is found at the top of HashMap.h. This is because the iterator is closely associated with the HashMap.

You must NOT modify the code in Main.cpp; it is the driver to test the app.

Hints

The Hash-Map must use a "Chained-Hashing" approach (text section 3.2). The base array is dynamically allocated and is of size 500. The constructor for HashMap will need to dynamically allocate this array and then initialize every entry to a nullptr.

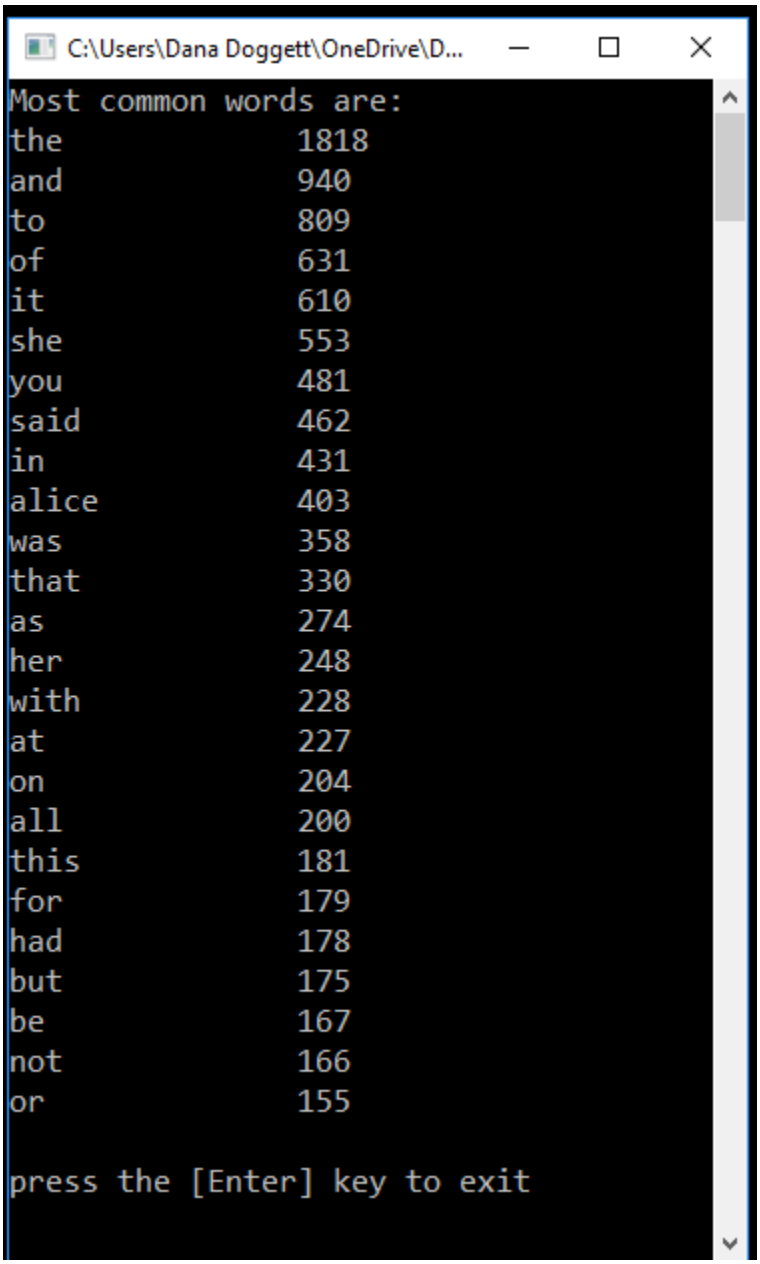
```
hashArray = new Node*[size];
for (int i = 0; i < size; i++)
{
        hashArray[i] = nullptr;
}
sizeOfArray = size;
totalElements = 0;
```

You MUST implement a destructor for the HaskMap. It will need to loop through the entire array and for every non-nullptr entry, delete each Node in the linked list. Once all the nodes have been deleted, you will have to also delete the dynamic array.

The Iterator part is somewhat tricky as well. Try to visualize on a white-board how the map is iterated. If you can't do it on a white-board, you will not be able to implement it.

Output

The results should look like this:



Turn In:

HashMap.cpp
Iterator.cpp

Grading

110 points

Documentation (5 pts)

Student name, Section and Disclaimer	_____/5

Hash Map (55 pts)

Constructor	_____/10
Destructor	_____/10
IsKeyPresent	_____/5
SetKeyValue	_____/10
GetKeyValue	_____/5
GenerateHash	_____/5
begin	_____/5
end	_____/5

Iterator (35 pts)

Constructor(s)	_____/5
`Iterator::operator++`	_____/20
`Iterator:: operator!=(Iterator & other)`	_____/5
`Iterator:: operator*()`	_____/5

Output (20 pts)

output looks like the sample output from above	_____/20