COSE213 Data Structures (Spring 2022)
Instructor: Prof. Won-Ki Jeong
Due date: May 15, 2022, 11:59 pm.

Assignment 3: Word frequency checker (v1.2)

In this assignment, you will implement map data structure to check the frequency of word occurrence in a text file. A Map is an abstract data structure that stores key-value (k,v) pairs. There are no duplicate keys in a Map. Your program reads in a text file, parse the text into words, and build a map to calculate the frequency of each word occurring in the given text. That means, the key is an input word (string) and the value is the number of occurrence (int) of that word in the text. Below is a simple example.

Input text:

```
I like a coffee and you like a tea.
We like a coffee and a tea.
```

Output word frequency:

```
a:4
like:3
tea:2
and:2
coffee:2
we:1
you:1
i:1
```

Note that, in this assignment, we convert each word to lower-case letters and ignore non-alphabetic characters to make the problem simpler (use non-alphabetical character as a delimiter, for example, "I've" should be decomposed into two words as "I" and "ve"). Note that "ve" in the previous example is not a correct word, but let's keep this way to make this assignment simpler.

To calculate the per-word occurrence count, we scan the input text word by word and count how many times each word appears. Hash map data structure is useful for this task. We can create a word-count pair for newly appeared word with 1 count, and if that word appears again later, increase the count by one.

1. Hash Map data structure (50 pts)

Hash Map is a template class defined as below:

```
template <class KeyType, class ValType>
class MapElem
{
public:
     typedef KeyType ktype;
     typedef ValType vtype;

     KeyType key;
     ValType val;

     MapElem* link;
};

template <class HashMapElemType>
class HashMap
{
     public:
          HashMap();
          ~HashMap();
     private:
          ...
};
```

Hash Map class should provide the functions below:

`HashMap(unsigned int c)`
Constructor with initial hash table size of c.

`int size()`
Return the size of the key-value pair

`bool isEmpty()`
Return true if no key-value pair is in the map

`HashMapElem* find(const key k)`
If k is in the map, return the pair (k,v) as MapElem. If find is not successful, return NULL.

`void insert(const key k, const value v)`

Insert a pair (k, v) in the map. If k is already in the map, change its value with v.

`bool remove(const key k)`
If k is in the map, remove its pair (k,v) and return true. If k is not in the map, return false.

`void print()`

Print all key:value pair in <u>decreasing order of value</u>.

`unsigned int hashfunction(const KeyType k)`

Returns an integer hash value converted from a key k.

You need to build a hash map with a given word as a key. You are required to implement the static hashing with the *<u>division hash function</u>* coupled with *<u>chaining</u>* for overflow handling. For a given key, you can convert it to an integer number using the algorithm discussed in the class.

For this assignment, you may assume that KeyType is string, and ValType is either int or float.

2. Checking word frequency (50 pts)

In `class WordFrequency`, you need to implement at least the following functions:

`void ReadText(const char* filename)`
Read in the input text file, parse it into words, and build a word-frequency map. You may use the following code snippet for parsing input string.

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

// s : input string
istringstream iss(s);

do
{
    string sub;
    iss >> sub;
    cout << "Substring: " << sub << endl;
} while (iss);
```

To convert each string to lower-case, you can use the following code snippet:

```
#include <algorithm>
#include <string>

std::string data = "Hello";
std::transform(data.begin(), data.end(),
               data.begin(), ::tolower);
```

`int GetFrequency(const std::string word)`
Find the frequency of a given word. If the word is not found, return 0.

`void IncreaseFrequency(const std::string word)`
Find the frequency of a given word and increase it by 1. If the word is not found, set its frequency by 1.

`void PrintAllFrequency()`
Print out the frequency of all words in the text in decreasing order of its occurrence (most often occurred word should be printed out first) as shown in the example of this handout. It can be simply call print() function of the hash map class.

If your implementation is correct, the skeleton main.cpp will print out the result for "steve_jobs.txt" as follows:

```
Print all results
the:98
i:93
to:71
and:67
it:55
was:48
a:46
of:42
that:39
in:35
you:35
my:30
```

```
is:29
had:22
out:20
...
```

3. Compile and submit

You must submit the code online via blackboard. You can compile the code as follows:

```
> make
```

The output executable name is assign_3. You can run your code by simply type in this name in the terminal.

```
> assign_3
```

You need to submit map.txx and wordfrequency.cpp only. When you submit your code, please change the filename to contain your student ID as follows:

```
2021XXXXXX_map.txx
2021XXXXXX_wordfrequency.cpp.
```

Good luck and have fun!