

## Assignment 2 – Word Counter

*Due: Wednesday, April 10<sup>th</sup>, 2019 by 11:55PM*

### 1. Introduction

In this assignment, you will use AVL data structure from the lecture note to implement a program to count words from a text file. The implementation of the assignment is organised into one `main()` function and three classes: `AVLNode`, `DynamicArray`, and `MapSet` with file separate files, namely, `AVLNode.h`, `DynamicArray.h`, `MapSet.h`, `MapSet.cpp` and `main.cpp`. The detail of these files are as follows:

#### a. `AVLNode.h`

This file contains the declaration and definition of `AVLNode` data structure.

```
class AVLNode {
public:
    string word;           // data
    DynamicArray list;     // data
    int freq;              // data
    AVLNode* left;        // left child
    AVLNode* right;       // right child
    int balance;          // balance factor
};
```

#### b. `DynamicArray.h`

The file contains the simplified version of `DynamicArray` data structure and the definition of necessary methods in this class. The prototype of this class is as following.

```
class DynamicArray {
private:
    int size;
    int capacity;
    int *storage;

public:
    DynamicArray(int capacity = 10);
    ~DynamicArray();

    void setCapacity(int newCapacity);
    void ensureCapacity(int minCapacity);
    void print();
    void add(int line);
};
```

#### c. `MapSet.h`

This file contains the declaration of `MapSet` data structure which is based on AVL tree introduced in the lecture note.

#### d. `MapSet.cpp`

This file contains the prototype for the method `add(string word, int line)` you need to implement in this assignment. The detail descriptions for all these methods are provided in the next section.

e. `main.cpp`

This file contains the `main()` function to test your implementation of the MapSet. It sequentially reads the text from the `input.txt` and adds to the MapSet object. The `main()` function contains a simple parser to extract words from a string line.

## 2. Your Tasks

Your tasks in this assignment is to implement the MapSet data structure to store how many times a word appears in the text and in which lines it occurs. In particular, you need to complete the following method `void add(string word, int line)` provided in the file "`MapSet.cpp`" so that it does the following: After reading the entire input file, it should output the total number of words and a list of all the words that occur in the file. This list should be in **ALPHABETICAL** order. Next to each word, you should put the frequency of word appearance and a list of the line numbers on which the word occurs. The list of the line numbers contains **NO** duplicated values and is sorted in **ASCENDING** order. The initial code provides you the skeleton to help you to complete the requested data structure.

For example, the following input:

```
This file contains the declaration of the MapSet
data structure which is based on the AVL tree.
```

will has the corresponding output:

```
The total number of words: 15
-----
Word           |      Freq. | Line No.
-----
avl            |           1 | 2
based          |           1 | 2
contains       |           1 | 1
data           |           1 | 2
declaration    |           1 | 1
file           |           1 | 1
is             |           1 | 2
mapset         |           1 | 1
of             |           1 | 1
on             |           1 | 2
structure      |           1 | 2
the            |           3 | 1 2
this           |           1 | 1
tree           |           1 | 2
which          |           1 | 2
-----
```

### 3. Evaluation – How assignments are graded?

This assignment is graded using the mechanism of autotester. In specific, there are totally 100 testcases. A testcase is considered “passed” if your output is completely matched with the expected output (solution). The number of “passed” testcases will correspond to your grade for this assignment.

### 4. Submission

In this assignment, you are allowed to submit ONLY TWO files `MapSet.h` and `MapSet.cpp` which contains all your code. Therefore, you are NOT allowed to change the printing format and the code in the other files including `AVLNode.h`, `DynamicArray.h`, and `main.cpp`. In addition, you should make sure that no other debugging `printf` or `cout` left behind in your code.

Instruction:

- Select two files: " `MapSet.h` " and " `MapSet.cpp` ", compress into "src.zip". Please **DO NOT** put them into any subfolder.
- Access website <http://www.cse.hcmut.edu.vn/onlinejudge/> and sign in using your university account and password.
- Submit "src.zip" to the system .

There are total 20 testcases, each one corresponds to 5 points. You can check the result for each testcase whether it is pass or failed. You can submit 20 times per day. The timeout for each testcase is 3 seconds. Your final mark is based on the final submission.

The deadline for this assignment **Wednesday, April 10<sup>th</sup>, 2019 by 11:55PM.**