# EEM 480 Homework 4

You are asked to implement the functions of a hash-based document tracking. The classes you required are about document indexing, which enables to speed up the content search of documents. Like all search engines do, all documents in the internet is indexed and inserted to a database. Thus, whenever you search for a document including word(s), search engines can bring the documents which contain the word you are looking for in a fraction of milliseconds. Obviously, they do not perform the actual search operation in that moment, i.e. when you search for the word. Instead, they are performing the search operation when they are *indexing* the documents. Thus they already know which documents include which words or phrases and the time consuming search operation is shifted to the offline stage.

Indexing can be done in several ways. In real life, search engines use huge matrices which keep the relationships between the documents and the words. In this assignment, we will keep the information within a hash table. In this project you are required to realize a hash table using **open addressing** method in order to solve collusion. Here it is preferred **double hashing** in order to distribute clusters evenly on database. For each word also keep the frequency variable in order to track the number of occurrences of a word in text.

| Index | |
|---|---|
| 0 | unintelligible 1 |
| 1 | |
| 2 | except 3 |
| 3 | |
| 4 | |
| 5 | is 13 |
| 6 | |

.
.

| | |
|---|---|
| n-4 | |
| n-3 | was 3 |
| n-2 | |
| n-1 | |
| n | house-front 1 |

- This hash structure will be used to trace a text file. The program will get a path of a text file written in English. The program will trace each word and keep the number of occurrences of each word. All punctuation marks will be removed. (Ex. The boy, who has green hair is walking down the street. “boy” and “street” has to be isolated from comma or dot)
- Try hash table size of ~1000, ~5000, ~10000 Check the number of occurrences of collusion. (I will definitely check)
- Explain how you obtain key from word.
- Explain your Hash function using double hash to solve the collusion.

Here the Interface for your HW is given as :

```
public interface HW3_Interface {
    Integer GetHash(String mystring);
    void ReadFileandGenerateHash(String filename, int size);
    void DisplayResult(String Outputfile);
    void DisplayResult();
    void DisplayResultOrdered(String Outputfile);
    int showFrequency(String myword);
    String showMaxRepeatedWord();
    boolean checkWord(String myword);
    float TestEfficiency();
}
```

Here the functions and their explanations :

`Integer GetHash(String mystring); // generate an integer value (hash index) related to the input word. If collusion occurs the collusion has to be solved by double hash method.`

`void ReadFileandGenerateHash(String filename, int size); // Create the open address hash structure with the size given by the user. The file which contains a very long text will be parsed and during the parsing hash table must be modified by the words.`

`void DisplayResult(String Outputfile); // All the words in the text and their frequency has to be displayed in a text file.`

`void DisplayResultOrdered(String Outputfile); // All the words and in the text and their frequency has to be displayed in a text file in an ordered fashion. The most repeated words will be listed at the beginning and the least repeated words at the end`

`void DisplayResult(); // All the words in the text and their frequency has to be displayed on the screen.`

`int showFrequency(String myword); The frequency of myword in the text file will be given. If there is no myword in the text -1 must be returned.`

`String showMaxRepeatedWord(); // The most repeated word has to be returned.`

`boolean checkWord(String myword); // Checks whether myword is found in the text.`

`Integer TestEfficiency(); // Returns the number of collusions during parsing the file.`

Obviously, by using a hash table instead of a matrix, we are saving from memory compare to employing a matrix. However, we are going a little bit down in terms of performance since we have to compute the hash value and go over more than one words on a linked list while looking for just one word. That's the trade off!

Another important thing about your assignment is the implementation of your hash function. Main duty of the hash function is generating a digestion respect to the input data. In this project, the digestion becomes an index value for your table and the input data is the word that you are indexing. If your hash function is a loose one, the overlapping ratio of the words increase. Then it causes a bad distribution of the indexed words and increases the search time. So we would like to have a hash function such that distributes the input words in a finely manner as much as possible. Briefly, you had better to make a little research on hash functions in order to implement your's a good one.

Example (In text file)

Outside, even through the shut window-pane, the world looked cold. Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemed to be no colour in anything, except the posters that were plastered everywhere. The blackmoustachio'd face gazed down from every commanding corner. There was one on the house-front immediately opposite. BIG BROTHER IS WATCHING YOU, the caption said, while the dark eyes looked deep into Winston's own. Down at streetlevel another poster, torn at one corner, flapped fitfully in the wind, alternately covering and uncovering the single word INGSOC. In the far distance a helicopter skimmed down between the roofs, hovered for an instant like a bluebottle, and darted away again with a curving flight. It was the police patrol, snooping into people's windows. The patrols did not matter, however. Only the Thought Police mattered.

Behind Winston's back the voice from the telescreen was still babbling away about pig-iron and the overfulfilment of the Ninth Three-Year Plan. The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it, moreover, so long as he remained within the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even conceivable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live -- did live, from habit that became instinct -- in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinized.

Winston kept his back turned to the telescreen. It was safer, though, as he well knew, even a back can be revealing. A kilometre away the Ministry of Truth, his place of work, towered vast and white above the grimy landscape. This, he thought with a sort of vague distaste -- this was London, chief city of Airstrip One, itself the third most populous of the provinces of Oceania. He tried to squeeze out some childhood memory that should tell him whether London had always been quite like this. Were there always these vistas of rotting nineteenth-century houses, their sides shored up with baulks of timber, their windows patched with cardboard and their roofs with corrugated iron, their crazy garden walls sagging in all directions? And the bombed sites

where the plaster dust swirled in the air and the willow-herb straggled over the heaps of rubble; and the places where the bombs had cleared a larger patch and there had sprung up sordid colonies of wooden dwellings like chicken-houses? But it was no use, he could not remember: nothing remained of his childhood except a series of bright-lit tableaux occurring against no background and mostly unintelligible.

Lutfullah is not found in the text
except is found and number of occurrences is 3
There are 103 collusion occurred.
….

Rules for HW Submission

. You must write your HW in NetBeans environment.
. You must write a report with name "**Report_HW4.pdf**" explaining your HW (purpose, how did you solve it, algorithm etc.) and what you the environment you used (NetBeans, for example). The person who read your report can easily use the class you have written.
. Discuss the result you have obtained.
. Submission should be in the form of a zip/rar. When extracted, the result should be a single folder with the name "HW4".
. Do not forget to put your report into the zip/rar file.
. The name of your project will be "**Name_Surame_HW4**. e.g. *Lutfullah_Arici_HW4*. If you do not obey the rule I will not grade your homework.
. You must bundle your whole project folder into your HW4.zip file.
. If I extract your project file, then import to my environment and if it doesn't work, you will be graded on 30 not 85. (Double check. It saves life)
. Do HW by yourself. Be honest.
. **Don't even think cheating, you get full mark but minus!**

**Grading Criteria:**
Implement a class that has fields for all the data contained in one data set (i.e. an integer, a floating-point number, a character, and a string), and member functions that perform the required computation and output. Data fields of your class should be declared as "private" members of the class, so you will need to implement get() and set() functions to access them. Member functions that perform computation and output should be "public" members of the class. Your program should then be modified so that it uses this class to store the data in a data set and to generate the required output.

A fully working program is worth 85 points. A useful and descriptive report file is worth 10 points. Useful comments throughout your program are worth 5 points. Be neat, clear, easy to read syntax (i.e. following coding style guidelines)

**EEM 480 Coding Guidelines**

What follows is a list detailing standards to be followed whenever you are programming for EEM 480 - Variable and function names are to be descriptive of what the variable represents, or what the function does (i.e. “int numDataSets” instead of “int n”). - The first word in the name of a variable or function is to be written in all lowercase letters. The first letter of each subsequent word is to be capitalized. Special characters (underscores and the like) should not be used except for in special cases (i.e. “int someReallyLongVariableName”). - Names of classes follow the same rule, with the exception that the first letter of the first word is also capitalized (i.e. “class MyClass”). - Proper spacing and indentation should be used (see examples):

```
//GOOD
for(int loopCount = 0; loopCount < 10; loopCount++)
{
       if (loopCount >= 4) //countdown from 5 to 0
       {
              System.out.print( 9 – loopCount );
       }
}
//BAD
for(int loopCount=0;loopCount<10;loopCount++)
{
if(loopCount>=4) //countdown from 5 to 0
{
System.out.print( 9 – loopCount );
}
}
```

- Class data members should be declared as “private” members of the class unless there is a compelling reason to do otherwise (and occasionally, there will be). - Class function members should generally be declared as “public” members of the class (though there will be more frequent exceptions to this rule than to the above rule).

- Global variable declarations should be avoided except in rare instances (for example, having a global variable that specifies whether “debug” mode is on or off).

- Generally speaking, a function that does computation should produce no output (except possibly for debug output), and a function that generates output should perform little to no relevant computation. Instead, have the function that performs computation return data (or a data structure) that can then be passed to the function that generates your output. The purpose of this is to keep your modules as specific and reusable as possible.

- Every function in a program should include comments detailing, at a minimum, what the function does, what the parameters are, and what the return value is.

- In instances where a descriptive enough variable name cannot be found while still using a reasonable amount of characters, additional comments are to be used to explain what the variable is used for.

- Particularly confusing pieces of code are to be accompanied by additional comments explaining what the code is doing (if it’s confusing to you, as the person who wrote it, think how much worse it’s going to be for a complete stranger trying to just look at it and understand what’s going on).