**Purpose:**

The purpose of this project is to create a two-dimensional implementation of a hashmap and perform different actions on it. The project would be divided into four independent submissions. However, it is important that you accurately complete each submission.

**Introduction:**

Before we dive into the deliverables, it is important that you understand what a hashmap is. A hashmap is essentially a key and value pairing, where the key is used to locate the value associated with it. For example, if we have the following pairs:

(“Jake”, “TA for CMPSC131”)

(“Richard”, “TA for CMPSC465”)

Then the keys are Jake and Richard, where Jake points to TA for CMPSC131 and Richard points to TA for CMPSC465. Here, we can use Jake to find TA for CMPSC131.

This project would use the same idea. However, it would be implemented using a two-dimensional list.

**Submission-1:**

For this submission, you are expected to read any text file and populate a two-dimensional hashmap using that text file. The input file would contain different words, where each word would be your key. The values you associate with that key would be the number of occurrences of that word and locations of each of those occurrences.

Let’s explain it with an example. Suppose your text file is as follows:

Sam Mead teaches the course CMPSC131

CMPSC131 is an important course

Your output/hashmap would look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Value | | |  |  |
| Occurrences | X location of first occurrence | Y location of first occurrence | X location of second occurrence | Y location of second occurrence |
| “Sam” | 1 | 1 | 1 |  |  |
| “Mead” | 1 | 1 | 2 |  |  |
| “teaches” | 1 | 1 | 3 |  |  |
| “the” | 1 | 1 | 4 |  |  |
| “course” | 2 | 1 | 5 | 2 | 5 |
| “CMPSC131” | 2 | 1 | 6 | 2 | 1 |
| “is” | 1 | 2 | 2 |  |  |
| “an” | 1 | 2 | 3 |  |  |
| “important” | 1 | 2 | 4 |  |  |

From the hashmap above, the words “Sam”, “Mead”, “teaches”, etc. are your keys. Now, the key “Sam” is associated with the values: 1(Number of occurrences), 1(X location of first occurrence), 1(Y location of first occurrence). Here, X location is the row number of that word and Y location is the number of word in that row. For instance, Sam is on row 1, and it is the first word in that row. Similarly, “an” is on 2nd row, and it is the third word in that row.

It is important to note that each word could occur multiple times in the text file. If a word is repeated, you would **update** the occurrences. However, when it comes to the location of words, you would add the location of the occurrence to the hashmap instead of updating it. Therefore, if you have 3 occurrences of a word, you will have three locations for the word.

For submission 1, you are supposed to do the following:

* Read the text file attached below: *[We would attach a file here]*
* Create a 2D list.
* Create a function **populate()** that would populate the 2D list using these three functions: **add\_occurrence**, **update\_occurrence** and **add\_location**.
* Here, add\_occurrence would simply add first occurrence of the word. This function should only be called if you are reading that word for the first time.
* update\_occurrence would be called whenever the word already exists in the 2D list. This function should fetch the previous occurrence of that word and add one to it.
* add\_location would add the X location and Y location of the word into the 2D list. This function would be called for each occurrence of the word.
* Invoke populate().
* Print your 2D list after invocation.

As always, we expect you to write a main() for this submission as well.

**NOTE: Your code should work for any text file and any number of words.**

*Sample Output of the sample file should be attached after this.*