**Submission 2**

**Purpose:**

The purpose of this submission is to create and run multiple search operations on a hashmap.

**Submission-2:**

Now that you have a working code for hashmaps, your goal is to traverse the hashmap, and perform different actions on that. Here is what you are supposed to do:

* Create a function named **entire\_map(),** that goes over every key value pair in your hashmap and prints it.
* Create a function **get\_value(),** that takes in a key and returns the values associated with it. If the key doesn’t exist or if the key doesn’t have any value, print “Invalid Entry”.

Let’s suppose our hashmap is as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |  |  |

If you invoke get\_value() using “Mead”, your output should be 1, 1, 2.

* Write a function **get\_location()** that takes in a key and the number of occurrence. Your function would use the key and the number of occurrence to return the location of that occurrence.

In the example above, if we invoke it using “CMPSC131” and 2, your function should return 2 and 1. This means that your function returns the location of the 2nd occurrence as specified in the function invocation. Now, if you invoke it using “course” and 1, your function would return 1, 5.

However, if your hashmap is unable to find that occurrence, it should return -1, -1.

**Submission 3**

**Purpose:**

The purpose of this submission is to create and run multiple deletion operations on a hashmap.

**Submission 3:**

The goal of this submission is to traverse the hashmap and perform different actions on it. Here’s what you are supposed to do:

* Create a function named **delete\_hashmap(),** that takes your entire hashmap, goes over each key value pair and removes all of the keys and their corresponding values.
* Create a function **delete\_entry()**, that takes in a key and removes all values associated with that key. Let’s suppose, your hashmap is as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |  |  |

If you invoke delete\_entry() using **“course”**, your function would remove 2,1, 5, 2, 5 from the hashmap, and your new hashmap would be as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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” |  |  |  |  |  |
| “CMPSC131” | 2 | 1 | 6 | 2 | 1 |
| “is” | 1 | 2 | 2 |  |  |
| “an” | 1 | 2 | 3 |  |  |
| “important” | 1 | 2 | 4 |  |  |

* Create a function **delete\_location()**, which takes in a key and the number of occurrence as input. Within the body of the function, go to the entry associated with that key and remove the location of the occurrence. For example, in the hashmap above, if we invoke delete\_location() using “CMPSC131” and 1, your function would remove 1, 6 and subtract 1 from the number of occurrences as well. Your 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” |  |  |  |  |  |
| “CMPSC131” | 1 | 2 | 1 |  |  |
| “is” | 1 | 2 | 2 |  |  |
| “an” | 1 | 2 | 3 |  |  |
| “important” | 1 | 2 | 4 |  |  |

Note: Your functions should work for any hashmap and any key-value pair.

Note: You can assume that we are only making changes to the hashmap and not the original text file. Therefore, when we delete an occurrence from the hashmap, we don’t have to worry about updating the locations of other values.