Using the skeleton app provided in the ***/Interesting Stuff/Hash Tables – Open Addressing/***folder, complete the implementation of an *Open Addressing* collision resolution hash table.

Use the provided input files (100.data, 1000.data etc.) to feed your hash table.

Requirements:

* Implement all the operations as described in the skeleton
* At the end, after every element has been inserted, make sure to print out the maximum number of collisions which occurred.
* Also at the end compute the number of times the hash table was resized
* Play around with the hash function to see which one works better (less collisions)
* Play around with the fill factor and initial hash table size and complete the tables below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hash Function (char \* c, int i);** | **ISF \*** | **MFF \*\*** | **Max Collisions** | **Number of Resizes** |
| H1 | .25 | .60 |  |  |
| H1 | .20 | .75 |  |  |
| H1 | .20 | .80 |  |  |
| H1 | .25 | .83 |  |  |
| H1 | .25 | .85 |  |  |
| H1 | .50 | .90 |  |  |
| H2 | .25 | .60 |  |  |
| H2 | .20 | .75 |  |  |
| H2 | .20 | .80 |  |  |
| H2 | .25 | .83 |  |  |
| H2 | .25 | .85 |  |  |
| H2 | .50 | .90 |  |  |
| H3 | .25 | .60 |  |  |
| H3 | .20 | .75 |  |  |
| H3 | .20 | .80 |  |  |
| H3 | .25 | .83 |  |  |
| H3 | .25 | .85 |  |  |
| H3 | .50 | .90 |  |  |

\* ISF = Initial Size Factor

\*\* MFF = Max Fill Factor

Have other combinations in mind? Feel free to fill-up the table with more tries to see if any interesting results come up!

H1:

***int hashFunction(char \* content, int i)***

***{***

***int length = strlen(content);***

***int k, sum;***

***for (sum=0, k=0; k < length; k++)***

***{***

***sum += content[k];***

***}***

***return sum % size;***

***}***

H2 – H3 🡪 your choices!

* Try to improve each time