Using the skeleton app provided in the ***/Interesting Stuff/Hash Tables – Chaining/***folder, complete the implementation of a *Separate chaining with linked lists* 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, compute the standard deviation of all the buckets’ sizes. (see [here](http://en.wikipedia.org/wiki/Standard_deviation#Basic_examples) an example of how to compute this) – also, read up on what standard deviation is and try to figure out what a low/high value of such a measure would mean in our case.
* Try things out with a bad hash function first (a H0 – think about the worst hash function which could still somehow distribute entries in more than 1 bucket)

|  |  |  |
| --- | --- | --- |
| **Hash Function (char \* c, int i);** | **ISF \*** | **σ (standard deviation)** |
| H0 | .10 |  |
| H0 | .20 |  |
| H0 | .35 |  |
| H0 | .50 |  |
| H0 | .75 |  |
| H0 | .85 |  |
| H1 | .10 |  |
| H1 | .20 |  |
| H1 | .35 |  |
| H1 | .50 |  |
| H1 | .75 |  |
| H1 | .85 |  |
| H2 | .10 |  |
| H2 | .20 |  |
| H2 | .35 |  |
| H2 | .50 |  |
| H2 | .75 |  |
| H2 | .85 |  |
| H3 | .10 |  |
| H3 | .20 |  |
| H3 | .35 |  |
| H3 | .50 |  |
| H3 | .75 |  |
| H3 | .85 |  |

\* ISF = Initial Size Factor

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

H0: bad hash function!

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