

Assignment 1: Hash Tables

CS3D5A, Trinity College Dublin

Deadline: 17:30 30/10/2020

Grading: The assignment will be **graded during the lab hours** on 30/10/2020

Submission: **Submit via blackboard.** Include a separate .c file for each task, and the short assignment report in pdf, word, or text file, detailing your results for each part

Goals:

- Learn how to implement a hash table in C
- Consider how to choose a hash function
- Learn how to evaluate the performance of a hash table based on collisions
- Use a hash table to extract information from a dataset of historical figures

Task 1: Getting Started - 5 Marks

For the first part of this assignment you should write a hash table to store **the frequency** of names in an (unordered) list of Irish surnames. It will therefore use `char` arrays as keys and store `int`s as values.

Use the hash function `hash1` that adds the integer value of the `chars` in the string:

```
int hash(char* s) {
    int hash = 0;
    while(*s) {
        hash = (hash + *s);
        s++;
    }
    return (hash % ARRAY_SIZE);
}
```

You should use a linear probing strategy to handle collisions. For simplicity, you do not need to worry about a dynamically growing hash table. You can use an array to store each key (assuming that there is a known `MAX_STRING_SIZE`) and a fixed size for the hash table (`ARRAY_SIZE`) and focus on implementing hashing and linear probing.

Test your hash table by loading the test data provided in `names.csv`. Write a program that allows you to test the frequency of a given string.

NB: Load the data using a csv parser (yours from assignment 0 or the one in the solution on blackboard) (if that doesn't work for you, keep making progress by hardcoding the data in your code!).

Update your code to count the number of collisions obtained and display it.

Listing 1: Sample output -values might not be accurate!

`names.csv` loaded!

```
Capacity    : 59
Num Terms   : 42
Collisions  : 46
Load        : 0.71%
```

Enter term to get frequency or type "quit" to escape

```
>>> Synnot
Synnot 2
>>> bleb
bleb not in table
>>> quit
```

Task 1 Mark Allocations	
Correct implementation of hash table	4 marks
Take a string as input and print number of times string occurred in data	1 mark

Task 2: Choosing a hash function - 3 Marks

Now find a better hash function `hash2` for the data considered. Feel free to consult online resources. Note, the sample data is only a sample! Do not overfit your function to the sample provided, it should work well with any lists of Irish surnames. Justify your choice of hash function in your report (half a page max). Test your function on the sample data, indicate in your report how many collisions occur, is it better than the result from task 1?

Task 2 Mark Allocations	
Justify your choice of hashing algorithm	1 mark
Implement and evaluate your hashing algorithm	2 marks

Task 3: Twice the Fun - 2 Marks

Choose a new hashing function `hash3`, and use it to augment your solution to Task 2 such that it uses double hashing instead of linear probing. As with the previous task, you should report the number of collisions which occurred in your hash table. Use this to demonstrate the improvement of double hashing over linear probing. Document in the report why you chose this function.

Listing 2: Sample output. Input from `person.csv`

```
.. ././data/people.csv loaded!
Capacity    : 99991
Num Terms   : 86077
Collisions  : 179986
Load        : 0.861%
Enter term to get frequency or type "quit" to escape
>>> Maguire
Maguire 104
>>> Magwire
Magwire 141
>>> quit
```

Task 3 Mark Allocations	
Successful implementation of second hashing function	2 marks

Task 4: A More Interesting Application - 5 Marks

Hash tables have numerous applications in computer science. One domain where they can find great use is that of Information Retrieval. Search terms extracted from a collection of documents are used as the keys and lists of documents are stored as values in the table. When a user issues a query, the list of all documents which may be of interest to them can be rapidly retrieved, ranked and presented.

For this task you have been given a file containing a list of people. The data which you have been given is real data produced with great effort and expense by a number of Trinity historians. The people mentioned are all individuals who were in some way involved with the 1641 Irish rebellion. Learning about these people is of great interest to historians, but the challenging nature of working with 17th century data can make this difficult.

For this task you should expand (a copy of) your solution to the previous tasks. Instead of storing term counts at each index, you should store a list of people with a given surname. The keys of the hash table will be surnames. The values will be linked lists of people.

As before you should provide a way to search for information in the hash table. Allow a user to enter a surname and get a list of people with the given surname as a search result. Test on the truncated data and then on the full dataset.

Listing 3: Sample output.

```
.. /.. /data/people.csv loaded!
Capacity : 99991
Num Terms : 14963
Collisions : 2724
Load : 0.150%
Enter term to get frequency or type "quit" to escape
>>> Poulton
Person ID Deposition ID Surname Forename Age Person Type
3700 818191r164 Poulton Anthony 0 Mentioned
3678 818189r163 Poulton Anthony 0 Mentioned
3664 818187r162 Poulton Anthony 0 Mentioned
3576 818185r161 Poulton Anthony 0 Deponent
3428 818161r152 Poulton Anthony 0 Mentioned
716 819112r142 Poulton Anthony 0 Mentioned
>>> Lawles
Person ID Deposition ID Surname Forename Age Person Type
49837 823038r036 Lawles Phillip 0 Debtor
45499 812318r259 Lawles Elizabeth 0 Deponent
>>> quit
```

Task 4 Mark Allocations	
Alter (a copy of) your previous solution so that it now stores lists of people as values instead of word counts. This alteration should include freeing any dynamically allocated memory required for the lists.	3 marks
Provide a means to search across your hash table of people	2 marks

Extra Bits - Not Graded

- As previously mentioned, the people in this dataset are real people. Do a search for your family name and see if anything comes up. If you find someone you would like to read about, the text of the depositions is available at <https://1641.tcd.ie/>.
- You may have noticed that some people's names are spelled "wrong", e.g. Maguire spelled "Magwire". The English language was not standardised until some time in the mid-18th century. What you are seeing is the early Anglicization of Irish names coupled with the fact that the rules of the English language were not yet well established. Can you think of a way to provide more tolerant search across the hash table so that people searching for "Maguire" might also get results for "Magwire"?