CCDSALG/GDDASGO Term 1, AY 2024 – 2025

Project 2 Documentation –Application of Hash Table Data Structure

DECLARATION OF INTELLECTUAL HONESTY / ORIGINAL WORK

We declare that the project that we are submitting is the product of our own work. No part of our work was copied from any source, and that no part was shared with another person outside of our group. We also declare that each member cooperated and contributed to the project as indicated in the table below.

Section	Names and Signatures	Task 1	Task 2	Task 3	Task 4,
<S11>	Barlaan, Benjamin	✔	✔		
<S11>	Jocson, Vince		✔	✔	
<S11>	Vergara, Ram			✔	✔

Fill-up the table above. For the tasks, put an 'X' or check mark if you have performed the specified task. Please refer to the project specifications for the description of each task. Don't forget to affix your e-signature after your first name.

1. FILE SUBMISSION CHECKLIST: put a check mark as specified in the 3rd column of the table below. Please make sure that you use the same file names and that you encoded the appropriate file contents.

FILE	DESCRIPTION	Put a check mark ✔ below to indicate that you submitted a required file
`hash.h`	header file for hash function, etc.	✔
`hash.c`	C source file for hash function, etc.	✔
`main.c`	main module	✔
`INPUT1.TXT` to `INPUT5.TXT`	5 sample input files (with increasing values of *n*)	✔
`OUTPUT1.TXT` to `OUTPUT5.TXT`	5 sample corresponding output files	✔
`GROUPNUMBER.PDF`	The PDF file of this document	✔

2. Indicate how to compile your source files, and how to RUN your exe files from the COMMAND LINE. Examples are shown below highlighted in yellow. Replace them accordingly. Make sure that all your group members test what you typed below because I will follow them verbatim. I will initially test your solution using a sample input text file that you submitted. Thereafter, I will run it again using my own test data:

- How to compile from the command line
  C:\MCO> gcc -Wall main.c -o main.exe

- How to run from command line
  C:\MCO>main

Next, answer the following questions:

a. Is there a compilation (syntax error) in your codes? (YES or NO). __NO__

WARNING: the project will automatically be graded with a score of **0** if there is syntax error in any of the submitted source code files. Please make sure that your submission does not have a syntax error.

b. Is there any compilation warning in your codes? (YES or NO) __NO___
WARNING: there will be a 1 point deduction for every unique compiler warning. Please make sure that your submission does not have a compiler warning.

3. Please indicate if you created your own original hash function or if you used a hash function from some reference material: __Based on standard hashing methods and DJB2_____

If you created your own original hash function: affix your signature on the given space

We honestly swear that we (the group members) created our own original hash function for MCO2 which is described as follows:

<provide a concise description or code of the hash function here>

Name and Signature: _____
Name and Signature: _____
Name and Signature: _____

If the

The hash function is described in __Standard Hashing methods and DJB2_____
(indicate the source or copy/paste a URL)

DJB2 was the inspiration for our hash table:

```
unsigned int hash_function(const char* str, int tablesize) {
   unsigned int hash = 0;
   for (int i = 0; str[i] != '\0'; i++) {
      hash = (hash * 37 + str[i]); //adds every character value to hash and multiplies by a prime number 37
   }// we use 37 as the prime instead of the usual 31

   return hash % tablesize; //reduces it to fit the table size at the END
}
```

hash function is from some reference material: https://theartincode.stanis.me/008-djb2/

4. Specify what collision resolution technique you implemented in your MCO2 (i.e., it is linear probing, quadratic probing or double hashing)? Linear Probing

5. Disclose **IN DETAIL** what is/are NOT working correctly in your solution. Please be honest about this. NON-DISCLOSURE will result in severe point deduction. Explain briefly the reason why your group was not able to make it work.

For example:

The following are NOT working (buggy):

a.

b.

We were not able to make them work because:

a.

b.

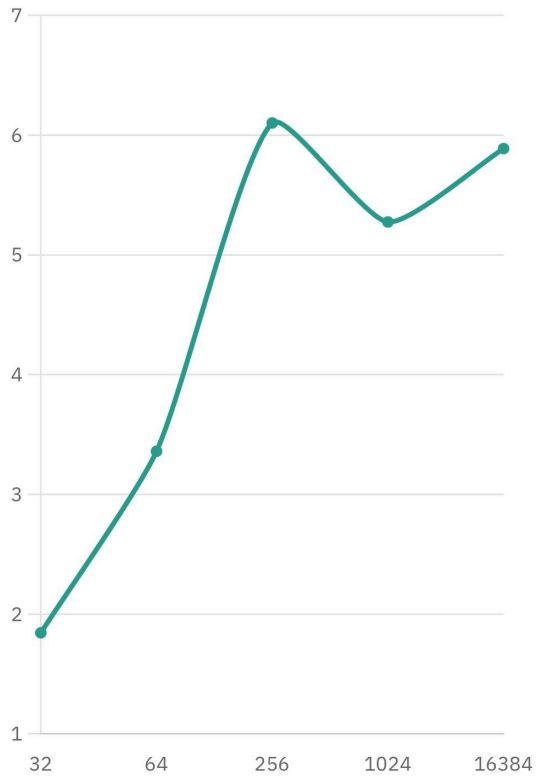
6. Based on the exhaustive testing that you did, fill-up the Table below. Test for at least 5 different values of *n*, starting with *n = 2^6 = 64* points. Set the values of *n* such that they are expressed using 2 as exponent. Your last test case should have *n = 2^14 = 16384 strings.*

Table: Statistics For the 5 Test Cases

Test Case #	n (input size)	# of keys/strings stored in the hash table	# of keys/strings stored in their home addresses	# of keys/ strings NOT stored in their home addresses	Average number of string comparisons
1	2^5 = 32	32	21	11	1.843750
2	2^6 = 64	64	28	36	3.359375
3	2^8 = 256	256	137	119	6.101563
4	2^10 = 1024	1024	578	446	5.274414
5	2^14 = 16384	16384	8937	7447	5.887207

NOTE: Make sure that you fill-up the table properly. It contributes 3 out of 15 points for the Documentation.

7. Create a line graph (for example using Excel) based on the Comparison Table that you filled-up above. The x-axis should be the values of *n* and the y axis should be the *average number of string comparisons*. Copy/paste an image of the graph below.



NOTE: Make sure that you provide a graph based on your comparison table data above. It contributes 3 out of 15 points for the Documentation.

8. Analysis – answer the following questions:

a. Based on the statistics above, is your Search() operation using a hash table slower or faster, on average, compared with linear search on an array? Do not simply answer with "slower" or "faster". You need to provide some explanation to support your answer.

Using a hash table is faster since using a linear search would have n number of string comparisons at the worst case while using the hash table only takes about 2-6 string comparisons on average based on our data.

b. Based on the statistics above, is your Search() operation using a hash table slower or faster, on average, compared with binary search on an array? Do not simply answer with "slower" or "faster". You need to provide some explanation to support your answer.

Using a hash table has a similar speed but still faster than binary search binary search has log n string comparisons, while the hash table has a lower number of search comparisons with it only increasing from 2 to 6 across levels of n^5 and n^12.

NOTE: Make sure that you provide cohesive answers to the questions above. This part contributes 3 out of 15 points for the Documentation.

9. Fill-up the table below. Refer to the rubric in the project specs. It is suggested that you do an individual self-assessment first. Thereafter, compute the average evaluation for your group, and encode it below.

REQUIREMENT	AVE. OF SELF-ASSESSMENT
1. Hash function	_30__ (max. 30 points)
2. Collision resolution function	__10_ (max. 10 points)
3. Search function	_15__ (max. 15 points)
4. Main module	_15__ (max. 15 points)
5. Input and output text files	_10__ (max. 10 points)
5. Documentation	15___ (max. 15 points)
6. Compliance with Instructions	_5__ (max. 5 points)

TOTAL SCORE __100_ over 100.

NOTE: The evaluation that the instructor will give is not necessarily going to be the same as what you indicated above. The self-assessment serves primarily as a guide.

サルバドル・フロランテ