100 Points

Due date:

11:59pm, Sunday 09/22/2019 for Monday labs.
11:59pm, Tuesday 09/24/2019 for Wednesday labs.
11:59pm, Thursday 09/26/2019 for Friday labs.

Purpose:

The purpose of this lab is to implement a hash table with closed hashing in C++.

General Requirements:

In this lab, you are required to implement a hash table with quadratic probing and double hashing using an array of structures (not arrays in STL).

The data structure is supposed to look like Yelp. Each Element is comprised of the Restaurant Name, its Rating, and its Price Range, where rating is an integer from 1-5, where 1 is worst and 5 is best. $$$ indicates the expensive segment, $$ indicates medium-priced restaurants, and $ indicates the cheapest restaurants. There will be only three segments for the expense: ‘$$$', ‘$$’, and ‘$’.

You are required to build two different hash tables for both of the hashing schemes, respectively. Each hash table will be an Array of Structures. You will need to show the hash table with each of these schemes separately. You are to read in a collection of strings and integers from a data file. All chars will be in lowercase, and your program can expect the user will always type chars in lowercase. Put the initial size of the hash table as 5. The data.txt file will contain the name of the restaurant[character], rating[integer from 1 through 5], and price[$ or $$ or $$$].

Your data.txt file may look like either of the below mentioned examples:

Example 1:
mcdonalds, 3, $
chipotle, 4, $$
pizzahut, 5, $$$
starbucks, 4, $$$

Or,

Example 2:
mcdonalds, 3, $, chipotle, 4, $$, pizzahut, 5, $$$, starbucks, 4, $$$

Choose the layout which is more convenient for you to read the file.
Only the restaurant’s name will be used as the key for calculating the hash value with the hash function. There shouldn’t be any duplicate keys inserted into the hash table. The file of strings you read from will be named data.txt. You may hard code the file name if you wish. After everything has been read from data.txt into the hash tables (which are Array of Structures), your program should ask the user to choose one of the options below:

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByRating
4- SearchByPrice
5- Print
6- Exit

Hash:

The hash table should implement an appropriate constructor and destructor. The rest of the methods should be implemented as follows:

- **Insert(name, rating, price)** – should insert parameters into both the hash tables (maintained by quadratic probing and the double hashing schemes) when it is not already present. Insertion must be done at the location which is obtained by the hashing function. If the location is full, then a new location will be calculated with quadratic probing and double hashing, respectively. Two different messages may be printed. The first message that will be printed will report on the success/failure of the insertion with quadratic probing. (The message in this case may look like: “[Name] has been successfully added at location[index x] using Quadratic Probing.”) Similarly, the second one will report on the success/failure of the insertion with respect to double hashing. (For example, “[Name] has been inserted at location[index y] using Double Hashing.”) Altogether, there can be four different cases like success-success, success-failure, failure-success, and failure-failure, all of which should be handled by the two messages shown as output.

- **Delete(name)** – should delete the key (i.e., name) and related values (i.e., rating and price) from both of the hash tables. Again, two different messages may be printed. The first message that will be printed will report on the success/failure of the deletion with quadratic probing. (The message in this case may look like: “[Name] has been successfully deleted from Quadratic Probing table.”) Similarly, the second one will report on the success/failure of the deletion with respect to double hashing. (For example, “[Name] has been deleted from Double Hashing table.”) Altogether, there can be four different cases like success-success, success-failure, failure-success and failure-failure, all of which should be handled by the two messages shown as output.

- **Print()** – should print out all of the elements of both of the hash tables one by one. Display the output of Quadratic Probing and Double Hashing in the format given below:
EECS 560 Lab 3 – Implementation of Closed Hashing
Prof.: Dr. Shontz, GTA: Chiranjeevi Pippalla, Anubhav Ghosh

<OUTPUT FOR QUADRATIC PROBING>

Index[0]: Name, Rating, Price
Index[1]: Name, Rating, Price
...
Index[m-1]: Name, Rating, Price

<OUTPUT FOR DOUBLE HASHING>

Index[0]: Name, Rating, Price
Index[1]: Name, Rating, Price
..........
Index[m-1]: Name, Rating, Price

• FindByName(name) – Two different messages will be printed. The first message that will be printed will report on the success/failure of the search by name operation on the quadratic probing hash table. (The message in this case may look like: “[Name] has been found at location [index x] of the Quadratic Probing hash table.”) Similarly, the second one will report on the success/failure of the search by name operation on the double hashing table. (For example, “[Name] has been found at location [index y] of the Double Hashing hash table.”) Altogether, there can be four different cases like success-success, success-failure, failure-success and failure-failure, all of which should be handled by the two messages shown as output.

• FindByRating(rating) – Finds all the restaurants with the given rating. You should give the input as an integer (rating), and the function should return two separate lists of all the restaurants having that rating in the two Hash Tables, respectively. Please take a look in the output patterns shown below in the ‘Expected Results’ section to get a clear picture. If the input is an invalid input or an input which is not an integer from 1 to 5, it should return “Invalid input”.

• SearchByPrice(price) – Finds all the restaurants with the given price range. You should give the input as a string of $ signs, and the function should return two separate lists of all the restaurants having that rating in the two Hash Tables, respectively. Please take a look in the output patterns shown below in the ‘Expected Results’ section to get a clear picture. For instance, if the user gives an input of ‘$$’, the function should return the restaurant with a price range of ‘$$’ from the hash table. If the input is invalid, it should return “Invalid input”.

• Hash function h(x) – convert each char of x into its ASCII value. The result of the ASCII value for each char of the string with its weight is summed and then given to the mod function with the size of the hash table. Finally, return the location of x as a result of applying the hash function.

If there is no collision, the index should be calculated as:

\[ h(x) = \text{(summation of ASCII values)} \]

\[ \text{Index} = h(x) \mod \text{table size} \]

Else, try resolving the collision using Quadratic Probing and Double Hashing as explained below:

Hashing with Quadratic probing:

\[ f_i = i^2, \forall i, 0 \leq i \leq k-1, \text{ where } x = \text{summation of ASCII values} \]
h₀(x) = (h(x) + 0²) mod table_size = h(x) mod table_size,
h₁(x) = (h(x) + f₁) mod table_size = (h(x) + 1²) mod table_size,
h₂(x) = (h(x) + f₂) mod table_size = (h(x) + 2²) mod table_size,
\ldots
h_{k-1}(x) = (h(x) + f_{k-1}) mod table_size = (h(x) + (k-1)²) mod table_size,
where fᵢ signifies i².

**Double Hashing:**

**h⁺ function:**

h⁺(x) = R - (x mod R), where R < table_size is a prime number.

Now, define fᵢ = ih⁺.

Observe that

h₀(x) = (h(x) + 0h⁺(x)) mod table_size = h(x) mod table_size,
h₁(x) = (h(x) + f₁) mod table_size = (h(x) + 1h⁺(x)) mod table_size,
h₂(x) = (h(x) + f₂) mod table_size = (h(x) + 2h⁺(x)) mod table_size,
\ldots
h_{k-1}(x) = (h(x) + f_{k-1}) mod table_size = (h(x) + (k-1)h⁺(x)) mod table_size.

**Expected Results:**

data.txt:

Already given in page 1.

As mentioned, 5 is the size of hash table. Hence, table_size = 5. Consider R = 3 for double hashing.

Note: The value of h(x) will be the result of summation of the ASCII value of each char from the input string.

*Note: The outputs shown below is only for the reference. The actual outputs may differ.

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

>6

Quadratic probing:

0: starbucks 4 $$$

12: mcdonalds 3 $

19: chipotle 4 $$

24: pizzahut 5 $$$

Double hashing:

10: starbucks 4 $$$$ 

11: chipotle 4 $$$

22: mcdonalds 3 $

25: pizzahut 5 $$$

.
Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

>1

Enter the data to be inserted:

>mcdonalds 4 $$$

Quadratic probing: mcdonalds is duplicated, can’t be added to the hash table.

Double hashing: mcdonalds is duplicated, can’t be added to the hash table.

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

>2

Enter a restaurant to be deleted:

> chipotle

Quadratic probing: chipotle is deleted from the hash table.

Double hashing: chipotle is deleted from the hash table.

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

>1

Enter a restaurant to be inserted:

> encore 3 $$$

Quadratic probing: encore is inserted into the hash table.
Double hashing: encore is inserted into the hash table.

------------------------------------------------------------

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

>6

Quadratic probing:

0: starbucks 4 $$$

. .

12: mcdonalds 3 $

. .

14: encore 3 $$$.

. .
Double hashing:
10: starbucks 4 $$$$

22: mcdonalds 3 $

25: pizzahut 5 $$$$

29: encore 3 $$$$

Please choose one of the following commands:

1- Insert
2- Delete
3- FindByName
4- FindByRating
5- SearchByPrice
6- Print
7- Exit

Enter the n rated restaurants: We have restaurants in five different categories of ratings 1 through 5, where 1 is the worst and 5 is the best. Please enter a rating between 1 and 5 to look for restaurants with that rating.

>6

6 star rated restaurants are not found in the hash table.

Please choose one of the following commands:
There are 5 possible ratings, i.e., 1 to 5 where 1 is the worst and 5 is the best. Please enter the rating for restaurants you want to find:

>5

Quadratic probing: pizzahut
Double hashing: pizzahut

Enter your price range:

>$$

Quadratic probing: mcdonalds
Double hashing: mcdonalds

Please choose one of the following commands:

1- Insert
Application: Bob is planning to develop an automated restaurant suggestion app like Yelp. This app should be able to recommend restaurants based on the factors of 'price' and 'ratings'. Your task is to answer the following questions:

1. Explain how this algorithm can help in application development.

2. Can you suggest a third factor which can be included in this kind of recommendation app? For example: the distance from your location.

Submission:

Follow the conventions below to facilitate grading:

Answers to the questions
• Make a file named “Answer_Lab3.pdf”, write your answers in that file, and submit it in PDF format in a separate subfolder. Also, list the paragraphs in sentences. Level: 1, aligned at 0.63 cm, and indented at: 1.27 cm.

Source Code

• Place all your source files (*.cpp, *.hpp) and input files in a single folder with no subfolders.
• Name your folder using the convention Lastname_LabX (e.g., Smith_Lab03).
• Include a functioning makefile inside the folder. (The makefile should also include the clean command.)
• Verify that your code runs on Linux.

Compressed File

• Compress using .zip, .rar, or .tar.gz.
• Name your file using the convention Lastname_LabX (e.g., Smith_Lab03.zip).

Email

• Use the following subject for your email: Lastname_LabX (e.g., Smith_Lab03).
• Send your code to chiranjeevi.pippalla@ku.edu if you are in one of Chiranjeevi’s sections or to anubhav@ku.edu if you are in one of Anubhav’s sections.
• Anytime you have a question about the lab, put the word question in the subject of the email.