Lab03

100 Points

Due date:
11:59pm, Monday 02/11/2019 for Tuesday labs.
11:59pm, Wednesday 02/13/2019 for Thursday 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 (not arrays in STL). You are required to build the hash table with both of the hashing schemes. You will need to show the hash table with each of these schemes separately. You are to read in a collection of strings from a data file of chars. **All chars will be in lowercase, and your program can expect the user will always type in chars in lowercase**. The first number from the data file is the size of the hash table. The rest of the file contains the **keys** that should be inserted in the hash table separated by a space. **There shouldn't be any duplicate keys inserted in the hash table**. The file of strings you read from will be **data.txt**. You may hard code the file name if you wish. After building the structure, 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- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit

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

- Insert(x) – should insert x 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. Display the message "x is added successfully" when x is inserted; otherwise display "Addition of x was not successful".
- Delete(x) – should delete the key x from both of the hash tables (maintained by quadratic probing and the double hashing scheme). The output should be either "x has been deleted" or "x is not found". Then continue to the next command.

- Print() – should print out all of the elements of both of the hash tables (i.e., those using quadratic probing and double hashing schemes) and should display output from bucket 0 to bucket m-1 in the format:

  Bucket 0: Element list
  Bucket 1: Element list
  …
  Bucket m-1: Element list

- Find(x) – output "x is found at location n" if the key x is in either one or both of the hash tables (maintained by quadratic probing and the double hashing scheme) and "x is not found" otherwise for both quadratic probing and double hashing, respectively. Also, if the input string is not present but, if it is present in reverse order in the hash tables then the message should be displayed "string is present in reverse order at location n" with both schemes.
- FindPalindrome – Find all palindrome strings from one of the hash tables. You should apply the hashing function starting from the beginning of the string and do the same from the end of the string. If the result of both hashing functions is equal, then return the string. Please note you can search for the palindrome string in any one of the hash tables (i.e., the one maintained by either quadratic probing or double hashing).
- ReverseString(x) – Reverse the input string present at location x in both the hash tables (maintained by quadratic probing and the double hashing scheme). If the location is empty, then display a message "Location x is empty".
- Hash function h(x) – **convert each char of x into its ASCII value for which only the first 8 chars are significant in x**. **Then apply weights to each char in x. This differentiates different strings composed from the same chars**. **The first char is assigned the weight of 10^0; the second char is assigned the weight of 10^1, and so on. The result of the ASCII value of each char of the string with its weight is added up and then given to the mod function with the size of hash table. Finally, return the location of x as a result of applying the hash function.**
- Rehash() – Rehash the table when the load factor, i.e., lambda, is greater than ½, and show the output of rehashing. Rehash should hash all the elements of an existing hash table into a new hash table, where the table size becomes the smallest prime number after 2*m for both of the tables maintained by quadratic probing and double hashing.

Index = summation of ASCII values % table size
Example1:
string A: computer
string B: computers
A and B are the same strings due to only consider the first 8 chars are significant

Example2:
string A: abc
string B: cba

index of A: (97*10^0 + 98*10^1 + 99*10^2) % table size
index of B: (99*10^0 + 98*10^1 + 97*10^2) % table size
Thus, string A and B will be placed into different bucket in hash table.

Hashing with Quadratic probing:

fi = i^2, $\forall i, 1 \leq i \leq k-1$, which is a family of quadratic functions, we have

$h_0(x) = (h(x) + 0^2) \bmod m$
$= h(x)$,
$h_1(x) = (h(x) + f_1) \bmod m$,
$= (h(x) + 1^2) \bmod m$,
$h_2(x) = (h(x) + f_2) \bmod m$,
$= (h(x) + 2^2) \bmod m$,
. . .
$h_{k-1}(x) = (h(x) + f_k) \bmod m$,
$= (h(x) + (k-1)^2) \bmod m$.

Double Hashing:

h^+ function:

$h^+(x) = R - (x \bmod R)$, where $R < m$ is a prime.

Now, define $f_i = ih^+$.
Observe that
$h_0(x) = (h(x) + 0h^+(x)) \bmod m$
$= h(x)$,
$h_1(x) = (h(x) + f_1) \bmod m$,
$= (h(x) + 1h^+(x)) \bmod m$,
$h_2(x) = (h(x) + f_2) \bmod m$,
$= (h(x) + 2h^+(x)) \bmod m$,
. . .
$h_k(x) = (h(x) + f_k) \bmod m$,
$= (h(x) + kh^+(x)) \bmod m$.

Expected Results:
data.txt: 17 epic is an adventure and famous creative poem

As mentioned, the first number 17 indicates the size of the hash table hence, **m = 17**. Consider **R = 5** for double hashing. Note: The value of x will be the result of calculation of the ASCII value of each char and its weight of the input string.

Please choose one of the following commands:

1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>6

Quadratic probing:

0: epic
2: adventure
5: poem
7: an
8: creative
11: and
13: famous
14: is

Double hashing:

0: epic
2: adventure
7: an
8: creative
11: and
13: famous
14: is
16: poem

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>1
Enter a string to be inserted:
>adventures
Quadratic probing: adventures is duplicated, can’t be added to the hash table

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

---
Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>2
Enter a string to be deleted:
> famous
Quadratic probing: famous is deleted from the hash table
Double hashing: famous is deleted from the hash table

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>2
Enter a string to be deleted:
> poem
Quadratic probing: poem is deleted from the hash table
Double hashing: poem is deleted from the hash table

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- Print
5- Exit
>1
Enter a string to be inserted:
> civic
Quadratic probing: civic is inserted into the hash table
Double hashing: civic is inserted into the hash table

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>1
Enter a string to be inserted:
> terret
Quadratic probing: terret is inserted into the hash table
Double hashing: terret is inserted into the hash table

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>6

Quadratic probing:

0: epic
2: adventure
7: an
8: creative
9: civic
11: and
14: is
16: terret

Double hashing:

0: epic
2: adventure
7: an
8: creative

9: civic
11: and
14: is
15: terret

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>3
Enter a string to be found:
>story
story is not found in the hash table.

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>3
Enter a string to be found:
>terret
Quadratic probing: terret is found at location 16
Double hashing: terret is found at location 15

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>4

Palindrome strings: civic terret

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
8- Print
9- Exit
>5
Enter location of string you want to reverse:
1
There is no string at location 1 with Quadratic probing.
There is no string at location 1 with Double hashing.

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
10- Print
11- Exit
>5
Enter location of string you want to reverse:
11
Quadratic probing: String “and” is changed to “dna”
Double hashing: String “and” is changed to “dna”

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>3
Enter a string to be found:
>and

Quadratic probing: “and” is present in reverse order at location 11.
Double hashing: “and” is present in reverse order at location 11.

Please choose one of the following commands:
1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>1
Enter string to be inserted:
computer

Rehashing…..

Quadratic probing:

13: an
16: terret
17: epic
19: adventure
21: civic
22: dna
26: creative
34: is
36: computer

Double hashing:

13: an
16: terret
17: epic
19: adventure
21: civic
22: dna
26: creative
34: is
36: computer

Please choose one of the following commands:

1- Insert
2- Delete
3- Find
4- FindPalindrome
5- ReverseString
6- Print
7- Exit
>7
Done!

Submission:
Follow the conventions below to facilitate grading:

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_Lab02).
- 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_Lab02.zip).

Email

- Use the following subject for your email: Lastname_LabX (e.g., Smith_Lab02).
- Send you code to l290w868@ku.edu if you are in one of Lei's sections or to dhwanipandya1401@ku.edu if you are in one of Dhwani's sections.
- Anytime you have a question about the lab, put the word question in the subject of the email.