Lab02

100 Points

Due date:
11:59pm, Monday 02/04/2019 for Tuesday labs.
11:59pm, Wednesday 02/06/2019 for Thursday labs.

Purpose:
The purpose of this lab is to implement a hash table with separate chaining (open hashing) in C++.

General Requirements:
For this lab, you are required to implement a hash table with separate chaining using a singly linked list. You are to read in the numbers from a data file of integers. 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 into the hash table. **There shouldn't be any duplicate keys inserted into the hash table. Let h(x_i) = x_i mod m be the hash function, where x_i is the key and m is the number of buckets**. The file of numbers 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-  Print
5-  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 the hash table when it is not found. Insertion must be done at the beginning of the chain. Output "x was added successfully" when x is inserted, otherwise output "x was not added successfully".
- Delete(x) – should remove the key x from the hash table, the output should be either "x has been deleted" or "x was not found" then continue to the next command.
- Print() – should print out all the elements of the hash table. Each chain must be separated by an arrow and ending with an endl. It should output from bucket 0 to bucket m-1 in the format:

  ```
  Bucket 0: Element list
  Bucket 1: Element list
  ```

…

Bucket m-1: Element list

- Hash(x) – **let h(x) = x mod m be the hash function, where x is the key and m is the number of buckets**. Return the location for key x.
- Rehash(x) – **Rehash the table when the load factor, i.e., lambda > 1 and output "hash table is rehashed". Rehash should hash all the elements of an existing hash table into a new hash table, which the table size becomes the smallest prime number after 2*m.**
- Find(x) – output "x is found" along with the location where it was found if the key x is in the hash table and "x is not found" otherwise.

Expected Results:
data.txt elements: 7 16 17 29 11 88 14 88

As mentioned, the first number 7 indicates the size of the hash table and it is chosen to be a prime.

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

0: -> 14
1: -> 29
2: -> 16
3: -> 17
4: -> 88 -> 11
5:
6:

----------------------------------------------------------
Please choose one of the following commands:
1-  Insert
2-  Delete
3-  Find
4-  Print
5-  Exit
>1
Enter a number to be inserted:
```

```
>38
38 is added to the hash table

-----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>1
Enter a number to be inserted:
>38
38 is a duplicate of 38, can’t be added to the hash table

-----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>2
Enter a number to be removed:
>16
16 is removed from the hash table

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

0: -> 14
1: -> 29
2:
3: -> 38 -> 17
4: -> 88 -> 11
5:
6:
```

```
----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>3
Enter a number to be found:
>100
100 can’t be found in the hash table.

----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>3
Enter a number to be found:
>14
14 is found at location 0.

----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>1
Enter a number to be inserted:
>1
1 is added to the hash table

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

```
Enter a number to be inserted:
>7
Hash table is rehashed.

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

0: -> 17
1: -> 1
2:
3: -> 88
4: -> 38
5:
6:
7: -> 7
8:
9:
10:
11: -> 11
12: -> 29
13:
14: -> 14
15:
16:

-----------------------------------------------------------
Please choose one of the following commands:
1-   Insert
2-   Delete
3-   Find
4-   Print
5-   Exit
>5
Bye bye!
```

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 the Linux lab machines.

Compressed File

- Compress the folder using .zip, .rar, or .tar.gz.
- Name your file using the convention Lastname_LabX (e.g., Smith_Lab02.zip).

Email

- Use Lastname_LabX (e.g., Smith_Lab02) as the subject line for your e-mail.
- Send your code to l290w868@ku.edu if you are in Lei's section and to dhwanipandya@ku.edu if you are in Dhwani's section.
- Anytime you have a question about the lab, put the word question in the subject of the email.