**CSE 4304-Data Structures Lab. Winter 2021**

**Lab-08**

**Date**: August 29, 2021 (Tuesday)

**Target Group:** All Lab groups

**Topic**: Hashing, Trie

**Instructions**:

- Task naming format: fullID_T01L08_1A.c/cpp
- If you find any issues in problem description/test cases, comment in the google classroom.
- Please comment if you find any tricky test cases that I didn’t include and others might forget to handle; please comment! I’ll be happy to add.
- I’ll mark the modified portions in BLUE color.

## Task-1:

Suppose you have a set of words stored in a Dictionary. Given a *prefix*, your task is to find out how many words start with it!

The first input line will be N, Q. N represents the number of words in the dictionary, and Q is the number of queries.
Your task is to print the number of words start with each corresponding prefix.
**Note**: Convert every string/prefix in lowercase before storing/ searching.

| **Sample Input** | **Sample Output** |
|---|---|
| 10 10 | |
| Beauty | |
| Beast | |
| Beautiful | |
| Amazing | |
| Amsterdam | |
| Beautify | |
| Banana | |
| Xray | |
| Beauty | |
| Glorifying | |
| A | 2 |
| Am | 2 |
| AM | 2 |
| Beauty | 2 |
| Beaut | 4 |
| Beast | 1 |
| Ing | 0 |
| AMS | 1 |
| Be | 5 |
| B | 6 |

**<u>Task-2</u>**

Implement the ‘Rabin-Karp String Matching Algorithm’ using the concept of the rolling-hash function. Test your program for different test cases. Make sure you understand how this algorithm is improving the traditional approach.

**<u>Task-3</u>**

Implement the following Collision handling techniques:

1. Linear Probing
2. Quadratic Probing
3. Double Hashing

The first input line should be (choice, N, Q), where ‘choice’ can be 1/2/3 corresponding to linear/quadratic/double hashing. N represents the size of the HashTable. Q represents the number of queries.
Then there will be Q numbers given as input.

Hash(x) = (x % TableSize)
Choose a suitable hash function for double hashing in addition to the function as mentioned above (as it needs two)

| **Sample Input** | **Sample Output** |
|---|---|
| 1 10 7 | |
| 35 | Inserted : Index-5 (L.F=.1) |
| 45 | Collision: Index-5 |
| | Inserted : Index-6 (L.F=.2) |
| 73 | Inserted : Index-3 (L.F=.3) |
| 36 | Inserted : Index-6 (L.F=.4) |
| 5 | Collision: Index-5 |
| | Collision: Index-6 |
| | Inserted : Index-7 (L.F=.5) |
| 24 | Inserted : Index-4 (L.F=.6) |
| 13 | Collision: Index-3 |
| | Collision: Index-4 |
| | Collision: Index-5 |
| | Collision: Index-6 |
| | Collision: Index-7 |
| | Inserted : Index-8 (L.F=.7) |

**Note**:

- L.F means Load Factor.
- If a number can’t be inserted within six attempts, abandon that number.

(Please test your program for different TableSize and different sets of numbers)