**CSE 4304-Data Structures Lab. Winter 23-24**

**Batch:** CSE 22

**Date**: November 13, 2024

**Target Group:** All

**Topic**: Hashing

**Instructions**:

* Regardless of how you finish the lab tasks, you must submit the solutions in Google Classroom. In case I forget to upload the tasks there, CR should contact me. The deadline will always be 11:59 PM on the day the lab took place.
* Task naming format: fullID\_T01L01\_2A.c/cpp
* If you find any issues in the problem description/test cases, comment in the Google Classroom.
* If you find any tricky test cases that I didn’t include but that others might forget to handle, please comment! I’ll be happy to add them.
* Use appropriate comments in your code. This will help you to recall the solution in the future easily.
* Obtained marks will vary based on the efficiency of the solution.
* Do not use <bits/stdc++.h> library.
* Modified sections will be marked with BLUE color.
* You are allowed to use the STL stack unless it’s specifically mentioned to use manual functions.

| **Group** | **Tasks** |
| --- | --- |
| 2A | 1 2 3 |
| 1B | 1 2 3 |
| 1A |  |
| 2B |  |
| **Assignments** | 2A/1B:  1A/2B: |

**Task-1**

Suppose you are implementing a Hash Table, but your hash function can’t guarantee to provide a unique index to each key. Hence you need to adopt some of the collision resolution techniques f(i).

Implement the following Collision handling techniques:

1. Linear Probing: f(i) = i
2. Quadratic Probing: f(i)=i^2
3. Double Hashing: f(i)=i\*hash2(x); hash2(x) = R - (x % R) with R=7

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.

* Main Hash function: Hash(x) = (x % TableSize)
* After each insertion, print the Load Factor (L.F.) representing the ratio between #number of inserted items and tableSize.

| **Sample Input** | **Sample Output** |
| --- | --- |
| 1 10 8  35  45  73  36  5  24  13  99 | Inserted : Index-5 (L.F=.1)  Collision: Index-5  Inserted : Index-6 (L.F=.2)  Inserted : Index-3 (L.F=.3)  Collision: Index-6  Inserted : Index-7 (L.F=.4)  Collision: Index-5  Collision: Index-6  Collision: Index-7  Inserted : Index-8 (L.F=.5)  Inserted : Index-4 (L.F=.6)  Collision: Index-3  Collision: Index-4  Collision: Index-5  Collision: Index-6  Collision: Index-7  Collision: Index-8  Input Abandoned  Inserted : Index-9 (L.F=.7) |
| 2 8 7  67  15  86  63  47  33  8 | Inserted : Index-3 (L.F = 0.125)  Inserted : Index-7 (L.F = 0.25)  Inserted : Index-6 (L.F = 0.375)  Collision: Index-7  Inserted : Index-0 (L.F = 0.5)  Collision: Index-7  Collision: Index-0  Collision: Index-3  Collision: Index-0  Collision: Index-7  Collision: Index-0  Input Abandoned  Inserted : Index-1 (L.F = 0.625)  Collision: Index-0  Collision: Index-1  Inserted : Index-4 (L.F = 0.75) |
| 3 15 11  94  46  61  29  85  77  46  63  67  93  61 | Inserted : Index-4 (L.F = 0.0666667)  Inserted : Index-1 (L.F = 0.133333)  Collision: Index-1  Inserted : Index-3 (L.F = 0.2)  Inserted : Index-14 (L.F = 0.266667)  Inserted : Index-10 (L.F = 0.333333)  Inserted : Index-2 (L.F = 0.4)  Collision: Index-1  Collision: Index-4  Inserted : Index-7 (L.F = 0.466667)  Collision: Index-3  Collision: Index-10  Collision: Index-2  Inserted : Index-9 (L.F = 0.533333)  Collision: Index-7  Collision: Index-10  Inserted : Index-13 (L.F = 0.6)  Collision: Index-3  Inserted : Index-8 (L.F = 0.666667)  Collision: Index-1  Collision: Index-3  Inserted : Index-5 (L.F = 0.733333) |

**Note**:

* If an item cant be inserted within six attempts, abandon that item.

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

**Task 2**

Given a collection of integers and a number ‘target’, find the pairs of integers whose summation is equal to ‘target’. The elements of the collection may not be unique.

The first line provides the collection of integers where -1 denotes the end of the input. The following line will contain the target value.

**Output:**

* Print the pairs whose summation equals ‘target’.
* If none of the pairs adds up to ‘target’, print ‘No pairs found’.

| **Sample Input** | **Sample Output** |
| --- | --- |
| 2 5 4 12 9 1 3 17 11 8 -1  13 | (8,5), (12,1), (9,4), (11,2) |
| 2 2 2 -1  4 | (2,2) |
| 2 5 4 2 0 1 3 -1  4 | (2,2), (3,1), (4,0) |
| 2 5 4 2 0 2 7 -1  6 | (4,2) |
| 2 5 4 12 9 1 3 17 8 11 8 5 -1  13 | (8,5), (12,1), (9,4), (11,2), (5,8) |
| 1 1 1 2 2 2 -1  3 | (1,2), (1,2), (1,2) |
| 1 1 1 1 2 2 1 1 1 1 2 -1  3 | (1,2), (1,2), (1,2) |
| 4 -2 2 7 9 1 3 1 0 -1  7 | (4,3), (7,0), (9,-2) |
| 2 5 4 12 9 1 3 17 11 8 10 -1  100 | No pairs found |

**Note**:

* Explore the ‘unordered\_map’ library function (<https://www.geeksforgeeks.org/unordered_map-in-cpp-stl/> )
* The elements/pairs can appear in any order. Hence [(8,5), (12,1), (9,4), (11,2)] and [(5,8), (1,12), (4,9), (2,11)] mean the same. There can be other combinations.
* **Rejected Solution:** Store the collection in an array. For every element, search the (target-current) element from the remaining portion of the array. Complexity O(n^2).
* Provide O(n) solution with Hashmaps

**Task 3:**

Given a sentence, you have to find the word(s) that occur more than once. Ignore the punctuation marks.

| **Input** | **Output** |
| --- | --- |
| data atad structure atad data | data 2  atad 2 |
| I know you know this, but you do not know of unknown trolls, because no known trolls will sew by windows, though they will owe you a hello when they throw a hoe, as it will go low and blow a hole in that window and so it will follow, that it happened awhile ago, as a troll will stand on a knoll and show you how to throw snow tomorrow at a rhino named Joe, who plays piano as he sips a cappucino and sings soprano in an inferno caused by a volcano in Reno with a casino at the bottom. Of the volcano. | volcano 2  a 9  by 2  will 5  at 2  throw 2  in 3  you 4  it 3  the 2  know 3  and 4  trolls 2  they 2  as 3  that 2 |
| I know you know this but you do not know of unknown trolls because no known trolls will sew by windows though they will owe you a hello when they throw a hoe as it will go low and blow a hole in that window and so it will follow that it happened awhile ago as a troll will stand on a knoll and show you how to throw snow tomorrow at a rhino named Joe who plays piano as he sips a cappucino and sings soprano in an inferno caused by a volcano in Reno with a casino at the bottom. Of the volcano | volcano 2  a 9  by 2  will 5  at 2  throw 2  in 3  you 4  it 3  the 2  know 3  and 4  trolls 2  they 2  as 3  that 2 |
| This refers to an exam where James had written ‘had had’ where John had written just ‘ had’. The examiner had approved James’ version. | James 2  where 2  had 6  written 2 |
| This refers to an exam where James had written had had where John had written just had The examiner had approved James version | James 2  where 2  had 6  written 2 |

**Note**:

* Show the words in any order.
* May use **getline**(cin, sentence) to take the input.