CSE 4304-Data Structures Lab. Winter 2024-25
Date: 28 January 2026
Target Group: All groups
Topic: Hash Table, Trie

Instructions:

- 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 recall the solution in the future easily.
- The 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**
1A/1B/2A/2B	**1 2 3 4**

Assignment	- Trie
	  - Mandatory: 5 6 8
	  - Optional: 7
	- Hashing
	  - Mandatory: 9 10 11
	  - Optional: 12

Task-1: Implementing a Basic Hash Table with Collision Resolution Strategy

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 using the function 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	 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

33 8	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 can't be inserted within six attempts, abandon that item.

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

Task 2: Find the target sum

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 add up to ‘target’, print ‘No pairs found’.

Sample Input	**Sample Output**	**Remarks**
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)	You can’t use an item more than once. So after using the (2,2) pair, only one item remains.
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)	Check this
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:

- Feel free to use STL for this task. 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 an O(n) solution with HashMaps (unordered map). Dont use STL Map as it'll increase complexity.

Task 3: Trie - Basic operations of the Trie data structure

Implement the basic operations of the 'Trie' data structure by implementing the following functions:

- void **insert**(): Inserts a string in a trie
- boolean **search**(): Returns whether the query string is a valid word.
- void **display**(): Shows all the words that are stored in the Trie in lexicographically sorted order.

- The first line of input contains space-separated words that need to be inserted in the Trie. Once the words are inserted, display all of them.
- The following line contains another collection of query words. Print **T/F** based on their presence/absence.

Sample Input	**Sample Output**
toy algo algorithm to tom also algea tommy toyota toy toyo al also algorithm algorithmic	algea algo algorithm also to tom tommy toy toyota T F F T T F

Task 4: Trie - Find the number of words starting with a certain prefix

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

The first input line will contain *N* and *Q*, where *N* represents the number of words in the dictionary, and Q is the number of queries. Print the number of words starting with each corresponding prefix.

Sample Input	**Sample Output**
10 10 Beauty Beast Beautiful Amazing Amsterdam Beautify Banana Xray Beauty Glorifying	
A	2
Am	2
AM	2
Beauty	1
Beaut	3
Beast	1
Ing	0
AMS	1
Be	4
B	5

Note: Convert every string/prefix to lowercase before storing/ searching. Don’t forget to handle duplicate entries.

Task 5: Trie - Search Suggestions

You are given a set of ‘products’ and a string ‘searchWord’. Design a solution that suggests **at most three** products after each character of searchWord is typed. Suggested products should have a common prefix with searchWord. If there are more than three products with a common prefix, follow the lexicographical order.

Input (products)	**Output**	**Explanation (searchWord)**
mobile mouse moneypot monitor mousepad mouse	mobile moneypot monitor mobile moneypot monitor mouse mousepad mouse mousepad mouse mousepad	**‘m’** **‘mo’** **‘mou’** (only 2 matches) **‘mous’** (only 2 matches) **‘mouse’** (only 2 matches)
havana havana	havana havana havana havana havana havana	**‘h’** **‘ha’** **‘hav’** **‘hava’** **‘havan’** **‘havana’**
juice jeerapani icecream jelly jam jackfruit jalapeno jeans	jackfruit jalapeno jam Jelly jeerapani Null Null Null	**‘J’**: 6 words matched. Printed only the first 3 in lexicographical order. **‘Je’**: 2 matches No match found for **‘jea’**, **‘jean’**, **‘jeans’**. Hence null.

Task 6: Trie - Max XOR value

You are given a collection of ‘n’ non-negative integers. Your task is to find the **maximum XOR value** between any two numbers in the collection. Implement an efficient solution to solve this problem.

Input	**Output**	**Explanation**
3 10 5 25 2 8	28	The maximum XOR is achieved between 5 (binary: 00101) and 25 (binary: 11001). The XOR result is 28 (binary: 11100).
0 1 2 3 4	7	Achieved between 3 (binary: 011) and 4 (binary: 100). The XOR result is 7 (binary: 111).
8 1 2 15	14	The maximum XOR is achieved between 1 (binary: 0001) and 15 (binary: 1111). The XOR result is 14 (binary: 1110).
1 1 1 1	0	All numbers are the same, so the XOR of any two numbers is 0.
5 25 10 2 8 12	29	
0	0	

Hint:

- Trie can be used to find solution in O(N)
- Each node in the Trie represents a bit (0 or 1) of the binary representation of numbers.
- Insert each number into the Trie bit by bit, from the most significant bit (MSB) to the least significant bit (LSB).
- Finding Maximum XOR:
    - For each number, traverse the Trie to find the number that gives the maximum XOR with the current number. This is done by attempting to match opposite bits (e.g., if the current bit is 0, look for 1).

Task 7: Trie - Can we segment the words?

Given a string and a set of words, return true if the string can be segmented into a sequence of one or more words from the set, separated by spaces.

Sample Input	**Sample Output**
court station food place -1 foodcourt	TRUE
foot hand ball -1 baseball	FALSE
cats dog sand and cat -1 catsandog	FALSE
apple orange pen pineapple banana -1 penpineappleapplepen	TRUE

Task 8: Trie - Does the abbreviation fit any word?

Given a list of words and an abbreviation, your task is to return ‘T’ (TRUE) if the abbreviation fits the words in that list. Otherwise, return F (FALSE).

Sample Input	**Sample Output**	**Explanation**
FooBar FooBarTest FootBall FrameBuffer ForceFeedBack -1 FB	T F T T F	FooBar / FootBall / FrameBuffer
FooBar FooBarTest FootBall FrameBuffer ForceFeedBack -1 FoBa	T F T F F	**Fo**oBar / **Fo**ot**Ba**ll
FooBar FooBarTest FootBall FrameBuffer ForceFeedBack -1 FoBaT	F T F F F	**FoBa**r**T**est

Task 9: Hashing - Words occurring more than once

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.

Task 10: Hashing - Morse Code

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:

- 'a' maps to ".-"
- 'b' maps to "-..."
- 'c' maps to "-.-." and so on.

For convenience, the full table for the 26 letters of the English alphabet is given below:

{".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."}

Given an array of strings words where each word can be written as a concatenation of the Morse code of each letter.

For example, "cab" can be written as "-.-..--...", which is the concatenation of "-.-.", ".-", and "-...". We will call such a concatenation the transformation of a word.
Return the number of different transformations along with the transformations among all words we have.

Sample Input	**Sample Output**
4 **gin** **zen** **gig** **msg**	**2** --...-. --...--.
6 **gin** **zen** **gig** **msg** **cab** **dog**	**4** --...-. --...--. -.-..--... -..-----.
1 **a**	**1** .-

Task 11: Hashing - Which words consist of the same unique character set?

Given a sentence and a word, your task is to print the words that consist of the same **unique character set**.

The first line of input will contain the sentence, and the second line will contain the word. Print the word(s) that consist of the same unique character set (length doesn't matter)

Input	**Output**
You may know the answer but it is not yam or maaayaaay or yammy may	may yam maayaaay yammy
student will act like students and it is studentish dont write studnet by mistake student	student students studnet
abcd abd acd bad baad baacd aabbccdd accc aadd abbe aaag aabbcd	abcd baacd aabbccdd
aceg aceegg ggccaaaee bcdg hbae aceg	aceg aceegg ggccaaaee

Note:

- Write a hash function which will check the similarity between two strings utilizing the unique characters. Just adding the ASCII of the unique characters won't work as 'ad' and 'bc' gives the same result. Need a better hash value.
- Explanation for input-1: The unique character set in 'may' is 'm,a,y'. As 'maaayaaay' also consists of the same set of unique characters 'm-a-y' it is a part of the output.
- All the unique characters of the word must be used. Hence, the word 'aabbcd'- 'abd', 'acd' etc is not a match as one of the unique characters are missing.
- The hash value of two words having the same unique character set should be equal.
- May use **getline**(cin, sentence) to take the input.

Task 12: Hashing - Babelfish

Problem Statement
You have just moved from Waterloo to a big city. The people here speak an incomprehensible dialect of a foreign language. Fortunately, you have a dictionary to help you understand them.

Input
Input consists of several dictionary entries, followed by a blank line, followed by a message consisting of many words. Each dictionary entry is a line containing an English word, followed by a space and a foreign language word. No foreign word appears more than once in the dictionary. The message is a sequence of words in the foreign language, one word on each line.

Output
Output is the message translated to English, one word per line. Foreign words not in the dictionary should be translated as **'eh'**.

Sample Test Case(s)

Input	**Output**
dog ogday cat atcay pig igpay froot ootfray loops oopslay atcay ittenkay oopslay	cat eh loops