

# CSN-261

## ASSIGNMENT-3

- Name-Kavya Barnwal
- Enrollment No.-18114039
- Course-B.Tech-CSE
- Email: [kbarnwal@cs.iitr.ac.in](mailto:kbarnwal@cs.iitr.ac.in)
- Contact:6205125517

\*=====\*

## **Problem Statement 1:**

Create a dictionary using Trie data structure (without using STL) having words and their meanings. You need to read the words and their respective meanings from a CSV file (uploaded in Piazza, named as TrieInput.csv), where 1st column is for words and 2nd column shows its meaning.

Given a word you have to print its meaning. If no such word is found in the dictionary, then print "Invalid word". Create a GUI using Qt library to accept a word in a text box and display the meaning in an another box, as shown in the Figure 1.

Also, create an installer of your program for Windows OS. You can use the software like

InstallSimple or InstallShield or WIX or NSIS to do so.

\*=====\*

## **Data Structure And Algorithm Used :**

### **Data Structures Used :**

Trie Data-Structure implemented for creating the dictionary

Arrays used to store all the child nodes of a given Trie - Node.

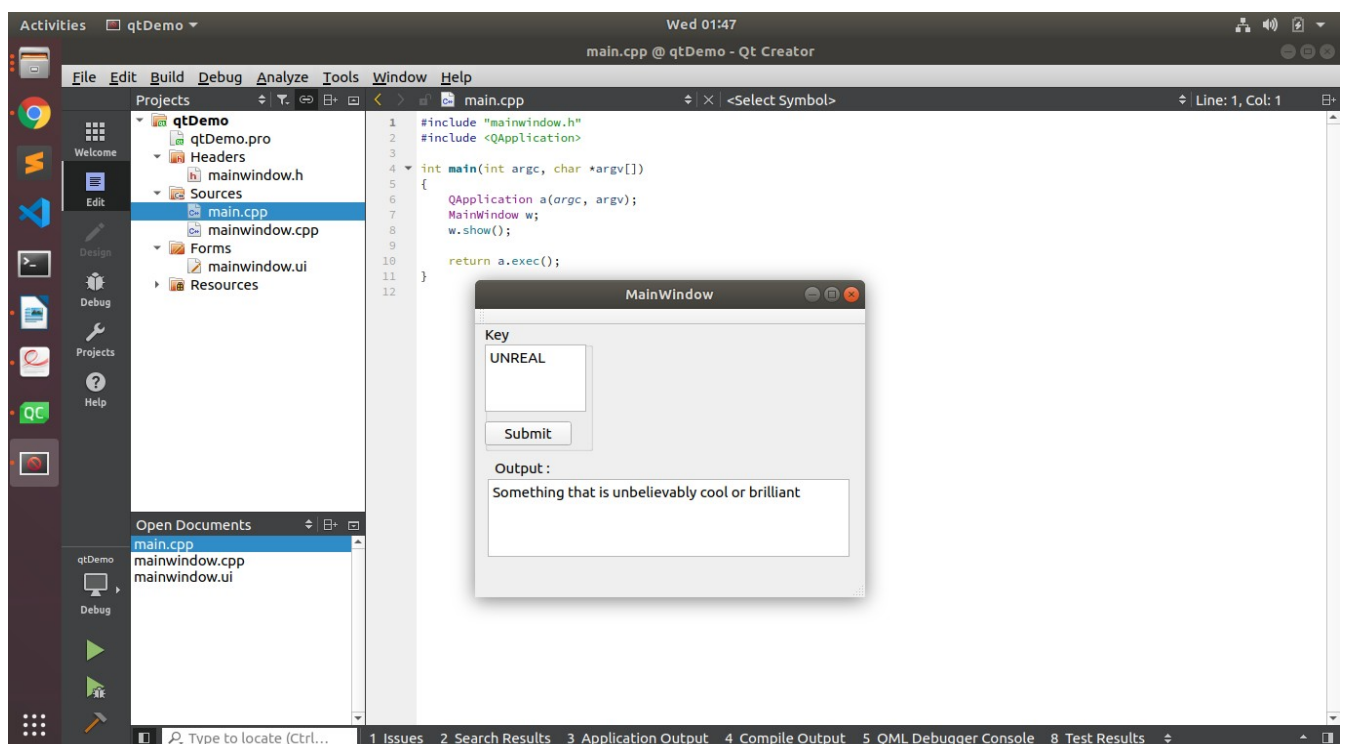
### **Algorithms Used :**

Inserting a key into Trie is a simple approach. Every character of the input key is inserted as an individual Trie node. Note that the *children* is an array of pointers (or references) to next level trie nodes. The key character acts as an index into the array *children*. If the input key is new or an extension of the existing key, we need to construct non-existing nodes of the key, and mark end of the word for the last node. If the input key is a prefix of the existing key in

Trie, we simply mark the last node of the key as the end of a word. The key length determines Trie depth.

Searching for a key is similar to insert operation, however, we only compare the characters and move down. The search can terminate due to the end of a string or lack of key in the trie. In the former case, if the *isEndofWord* field of the last node is true, then the key exists in the trie. In the second case, the search terminates without examining all the characters of the key, since the key is not present in the trie.

Figure for created GUI:



\*=====\*

## **Problem Statement 2:**

Implement N Queens problem to show all the possible combinations in N x N binary matrix and to display the total number of such combinations at the end, where 1 represents the position of N queens in the N x N matrix and remaining cells are represented by 0.

A sample output for N=4 is shown below.

Input: N: 4

Output:

Combination 1:

0 1 0 0

0 0 0 1

1 0 0 0

0 0 1 0

Combination 2:

0 0 1 0

1 0 0 0

0 0 0 1

0 1 0 0

Total number of combinations: 2

\*=====\*

## **Data Structure And Algorithm Used :**

### **Data Structures Used :**

2-d Arrays used to represent the arrangement of queens.

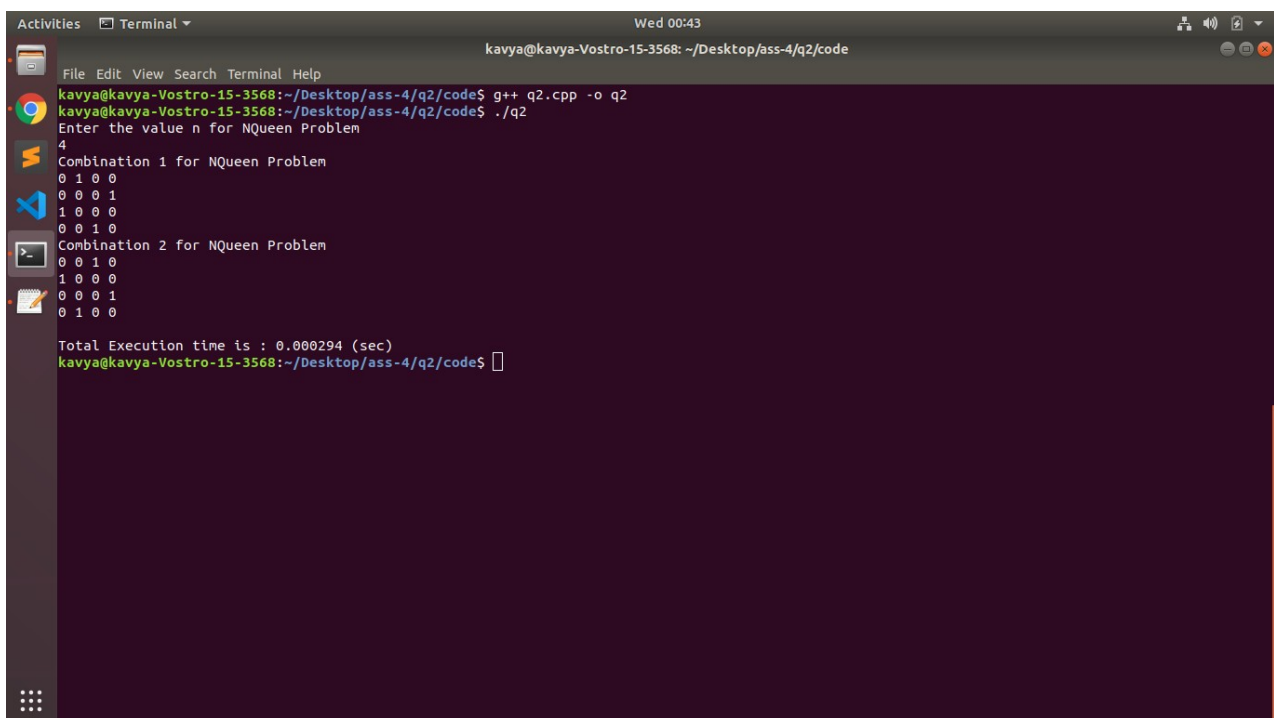
## Algorithms Used :

### **Backtracking Algorithm**

The idea is to place queens one by one in different rows,

starting from the lowermost row. When we place a queen in a row, we check for conflicts with already placed queens. In the current row, if we find a column for which there is no clash, we mark this row and column as part of the solution.

If we do not find such a row due to clashes then we backtrack and move forward returning false.



```
Activities Terminal
Wed 00:43
kavya@kavya-Vostro-15-3568: ~/Desktop/ass-4/q2/code
File Edit View Search Terminal Help
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q2/code$ g++ q2.cpp -o q2
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q2/code$ ./q2
Enter the value n for NQueen Problem
4
Combination 1 for NQueen Problem
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
Combination 2 for NQueen Problem
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
Total Execution time is : 0.000294 (sec)
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q2/code$
```

\*=====\*

### **Problem Statement 3:**

Given an integer array having N number of elements, write a C++ program using hash map (using STL) to find the length of the largest subarray from the given input array, where the summation of the elements of the subarray is equal to n. In the output, if any solution exists then print the starting and ending index (with respect to given input array) of the largest subarray and also print its length. Otherwise, print "Not Found", as described in the following output.

Input:

N = 8

15 0 2 -3 1 5 3 -2

n = 5

Output:

Length of longest subarray is 5

Index from 1 to 5

\*=====\*

### **Data Structure And Algorithm Used :**

#### **Data Structures Used :**

- Arrays used to store the given values.

- UnOrdered HashMap from c++ STL to store the accumulated sum upto a given index.

### **Algorithms Used :**

- Initialize sum = 0 and maxLen = 0.
- Create a hash table having (sum, index) tuples.
- For i = 0 to n-1, perform the following steps:

Accumulate arr[i] to sum.

If sum == k, update maxLen = i+1.

Check whether sum is present in the hash table or not. If not present, then add it to the hash table as (sum, i) pair.

Check if (sum-k) is present in the hash table or not. If present, then obtain index of (sum-k) from the hash table as index. Now check if maxLen < (i-index), then update maxLen = (i-index).

- Return maxLen.

```
Activities Terminal Wed 00:42
kavya@kavya-Vostro-15-3568: ~/Desktop/ass-4/q3/code
File Edit View Search Terminal Help
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$ g++ q3.cpp -o q3
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$ ./q3
Enter the number of elements for array
8
Enter elements
15 0 2 -3 1 5 3 -2
Enter k :
5
Length of longest subarray is 5
Index from 1 to 5
Total Execution time is : 0.000367 (sec)
kavya@kavya-Vostro-15-3568:~/Desktop/ass-4/q3/code$
```

\*=====\*