**Case studies on Dictionary**

**Case Study 1:**

**Sample Example on Dictionary**

Given  names and phone numbers, assemble a phone book that maps friends' names to their respective phone numbers. You will then be given an unknown number of names to query your phone book for. For each  queried, print the associated entry from your phone book on a new line in the form name=phoneNumber; if an entry for  is not found, print Not found instead.

**Note:** Your phone book should be a Dictionary/Map/HashMap data structure.

**Input Format**

The first line contains an integer, , denoting the number of entries in the phone book.  
Each of the  subsequent lines describes an entry in the form of  space-separated values on a single line. The first value is a friend's name, and the second value is an -digit phone number.

After the  lines of phone book entries, there are *an unknown number of lines of queries*. Each line (query) contains a  name to look up, and you must continue reading lines until there is no more input.

**Output Format**

On a new line for each query, print Not found if the name has no corresponding entry in the phone book; otherwise, print the full  and  in the format name=phoneNumber.

**Sample Input**

3

sam 99912222

tom 11122222

harry 12299933

sam

edward

harry

**Sample Output**

sam=99912222

Not found

harry=12299933

**Explanation**

We add the following  n=3 *(Key,Value)* pairs to our map so it looks like this:

phonebook = {(sam,99912222),(tom, 11122222),(harry,12299933)}

We then process each query and print key=value if the queried  is found in the map; otherwise, we print Not found.

*Query 0:* sam  
Sam is one of the keys in our dictionary, so we print sam=99912222.

*Query 1:* edward  
Edward is not one of the keys in our dictionary, so we print Not found.

*Query 2:* harry  
Harry is one of the keys in our dictionary, so we print harry=12299933.

Code:

#include <iostream>

#include <map>

using namespace std;

int main() {

std::map<string, string> phoneBook;

int n;

cin >> n;

// Read names and numbers, add to phoneBook:

for(int i = 0; i < n; i++){

string name;

string phone;

cin >> name;

cin >> phone;

phoneBook[name] = phone;

}

// Execute queries:

std::map<string,string>::iterator it;

string query;

while( cin >> query ){

it = phoneBook.find(query);

if ( it != phoneBook.end() ){ // key is found in phoneBook

cout << it->first << "=" << it->second << '\n';

}

else{ // the iterator hit the end of the phone book without finding key

cout << "Not found" << '\n';

}

}

return 0;

}

**Case Study 2:**

You have a weather forecast data having temperature details of few cities for few days for the year 2018

|  |  |  |
| --- | --- | --- |
| City | Date | Temperature |
| Delhi | 9-11-2018 | 45 |
| Bangalore | 9-11-2018 | 24 |
| Ranchi | 9-12-2018 | 28 |
| Chennai | 9-01-2018 | 38 |

Build data structure to answer the following queries

1. What is Temperature in Delhi on 9-11-2018
2. What is max temperature recorded in Chennai in 2018
3. Displaying the number of entries of 2018
4. Deleting the entry of Mumbai on 9-11-2018
5. Deleting all entries of Mumbai

Hint:

Inorder to create a dictionary of weather <city<date,temperature> ( Key= city , value<date,temperature)

Declaration in c++:

multimap<string,map<string,int>>