**#include** **<**cmath**>**

**#include** **<**cstdio**>**

**#include** **<**vector**>**

**#include** **<**iostream**>**

**#include** **<**algorithm**>**

**using** **namespace** std;

**int** addNumbers(**int** a, **int** b) {

**return** a+b;

}

**int** main() {

   string input = [

                ["Jamie", "city1", 1231321249],

                ["Dana", "city1", 1231321248],

                ["Jamie", "city2", 1231321245],

                ["Jamie", "city3", 1231321248],

                ["Dana", "city3", 1231321247],

                ["Jamie", "city4", 1231321247],

                ["Dana", "city4", 1231321245]

        ]

**for** (**int** i =0;i<input.size();i++)

    sort(input[0][i][2].begin(),input[0][i][2].end())

**return** input;

}

*// Your old code in python3 has been preserved below.*

*// """*

*// Suppose you have a big text log file and there are three columns:*

*// ------------------------------------*

*// Name    |  City     | Timestamp*

*// -------------------------------------*

*// Jamie      city1        ts4*

*// Dana       city1        ts3*

*// Jamie      city2        ts1*

*// Jamie      city3        ts3*

*// Dana       city3        ts2*

*// Jamie      city4        ts2*

*// Dana       city4        ts1*

*// ===============================*

*// The first column is a rider name, the second column is a city name, the final column is a timestamp (32 bit integer).*

*// This log file maintains a history of riders. We define a "triple" to be a tuple of three consecutive cities a rider has been to that are ordered by timestamp.*

*// Suppose t1<t2<t3<t4, then we will have the following triples:*

*// for Jamie, we have: (city2, city4, city3) , (city4, city3, city1)*

*// for Dana, we have: (city4, city3, city1)*

*// Write a program*

*// - to output all the triples in the file and*

*// - for each triple, output the count of unique riders that have been to that triple.*

*// For the example input, the output would be*

*// (city2, city4, city3) 1*

*// (city4, city3, city1) 2 ...*

*// sample input"*

*// input = Jamie,city1,1231321249*

*// Dana,city1,1231321248*

*// Jamie,city2,1231321245*

*// Jamie,city3,1231321248*

*// Dana,city3,1231321247*

*// Jamie,city4,1231321247*

*// Dana,city4,1231321245*

*// k = 1, x xlogx*

*// k = 2, y, ylogy*

*//     Jamie,city1,1, 1*

*//     Dana,city1,2, 1*

*//     Dana,city2,5, 2*

*//     Jamie,city2,3, 2*

*// group by name, list(city, timestamp)*

*//     Jamie,[(city1,1), (city2, 2)]*

*//     Dana, [(city2, 2), (city1,1)]*

*// """*

Amazon ER diagram

Top 5 products based on date

Codind round –

Design generic log parser framework

https://www.geeksforgeeks.org/minimize-sum-of-an-array-by-at-most-k-reductions/