Post Transition ★

24/25 challenges solved

Rank: 13110 | Points: 560

C language ★★★★★

Your Post Transition submission got 50.00 points. Share Tweet

Try the next challenge | Try a Random Challenge

Problem Submissions Leaderboard Editorial

We live in a big country. This country has $towns\_count$ towns in it. Each town has some post offices in which packages are stored and transferred.

Post offices have different inner structure. Specifically, each of them has some limitations on the packages it can store - their weight should be between min_weight and max_weight inclusively, where min_weight and max_weight are fixed for each office.

Packages are stored in some order in the office queue. That means, that they are processed using this order when sending and receiving.

Sometimes two post offices, even in different towns, may organize the following transaction: the first one sends all its packages to the second one. The second one accepts the packages that satisfy the weight condition for the second office and rejects all other ones. These rejected packages return to the first office back and are stored in the same order they were stored before they were sent. The accepted packages move to the tail of the second office's queue in the same order they were stored in the first office.

You should process several queries in your program. You'll be provided with structures $package$, $post\_office$ and $town$. in order to complete this task, you should fill the following functions:

$print\_all\_packages$ - given the town t, print all packages in this town. They should be printed as follows:

```
Town_name:
    0:
        id_0
        id_1
        ...
    1:
        id_2
        id_3
        ...
    ...
```

where 0, 1 etc are the numbers of post offices and id_0, id_1... are the ids of packages from the 0th post office in the order of its queue, id_2, id_3 are from the 1st one etc. There should be one '\t' symbol before post office numbers and two '\t' symbols before the ids.

$send\_all\_acceptable\_packages$ - given the towns $source$ and $target$ and post office indices $source\_office\_index$ and $target\_office\_index$, manage the transaction described above between the post office #$source\_office\_index$ in town $source$ and the post office #$target\_office\_index$ in town $target$.

$town\_with\_most\_packages$ - given all towns, find the one with the most number of packages in all post offices altogether. If there are several of them, find the first one from the collection $towns$.

$find\_town$ - given all towns and a string $name$, find the town with the name $name$. It's guaranteed that the town exists.

Input Format

First line of the input contains a single integer $towns\_count$. $towns\_count$ blocks follow, each describing a town.

Every town block contains several lines. On the first line there is a string $town\_name$ - the name of the town. On the second line there is an integer $offices\_count$ - the number of the offices in the town. $offices\_count$ blocks follow then, each describing an office.

Every office block also contains several lines. On the first line there are three integers separated by single spaces: $packages\_count$ (the number of packages in the office), min_weight and max_weight (described above). $packages\_count$ blocks follow, each describing a package.

Every package block contains exactly two lines. On the first line there is a string id which is an id of the package. On the second line there is an integer $weight$ which is the weight of the package.

Then, there is a single integer $queries$ on the line which is the number of queries. $queries$ blocks follow, each describing a query.

Every query block contains several lines. On the first line there is an integer 1, 2 or 3. If this integer is 1, on the second line there is a string $town\_name$ - the name of town for which all packages should be printed. If this integer is 2, on the second line there are string $source\_name$, integer $source\_office\_index$, string $target\_name$ and integer $target\_office\_index$ separated by single spaces. That means transactions between post office #$source\_office\_index$ in the town $source\_name$ and post office # $target\_office\_index$ in the town $target\_name$ should be processed.

If the integer is 3, no lines follow and the town with the most number of packages should be found.

Constraints

- All integer are between 0 and 10
- $towns\_count > 0$, $offices\_count > 0$.
- All strings have length ≤ 5
- All towns' names have only uppercase english letters and are unique.
- All packages' ids have only lowercase english letters and are unique.
- For each post office, $min\_weight \leq max\_weight$.
- All queries are valid, that means, towns with the given names always exist, post offices with the given indices always exist.

Output Format

For queries of type 1, print all packages in the format provided in the problem statement. For queries of type 3, print "Town with the most number of packages is $town_name$" on a separate line.

Sample Input 0

```
2
A
2
2 1 3
a 2
b 3
1 2 4
c 2
B
1
4 1 4
d 1
e 2
f 3
h 4
5
3

2 B 0 A 1
```

```
3
1 A
1 B
```

Sample Output 0

```
Town with the most number of packages is B
Town with the most number of packages is A
A:
    0:
        a
        b
    1:
        c
        e
        f
        h
B:
    0:
        d
```

Explanation 0

Before all queries, town B has 4 packages in total, town A has 3. But after transaction all packages from B's 0th post office go to the 1st post office of A, except package d because it's too light.

Change Theme Language: C

```c
        case 1:
            scanf("%s", town_name);
            town* t = find_town(towns, towns_count, town_name);
            print_all_packages(*t);
            break;
        case 2:
            scanf("%s", town_name);
            town* source = find_town(towns, towns_count, town_name);
            int source_index;
            scanf("%d", &source_index);
            scanf("%s", town_name);
            town* target = find_town(towns, towns_count, town_name);
            int target_index;
            scanf("%d", &target_index);
            send_all_acceptable_packages(source, source_index, target, target_index);
            break;
        case 3:
            printf("Town with the most number of packages is %s\n", town_with_most_packages(towns, towns_count).name);
            break;
        }
    }
    return 0;
}

```

Line: 158 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Upload Code as File

Run Code

Submit Code

You have earned 50.00 points!

24/25 challenges solved.

96%

C language

★★★★★

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
2
A
2
2 1 3
a 2
b 3
1 2 4
c 2
B
```

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature