#include <stdio.h>

#include <stdlib.h>

#define MAX\_STRING\_LENGTH 6

int i,j,k;

struct package

{

char\* id;

int weight;

};

typedef struct package package;

struct post\_office

{

int min\_weight;

int max\_weight;

package\* packages;

int packages\_count;

};

typedef struct post\_office post\_office;

struct town

{

char\* name;

post\_office\* offices;

int offices\_count;

};

typedef struct town town;

void print\_all\_packages(town t)

{

printf("%s:\n", t.name);

for ( i = 0; i < t.offices\_count; i++)

{

printf("\t%d:\n", i);

for ( j = 0; j < t.offices[i].packages\_count; j++)

printf("\t\t%s\n", t.offices[i].packages[j].id);

}

}

void send\_all\_acceptable\_packages(town\* source, int source\_office\_index, town\* target, int target\_office\_index)

{

int n = 0;

for ( i = 0; i < source->offices[source\_office\_index].packages\_count; i++)

if (source->offices[source\_office\_index].packages[i].weight >= target->offices[target\_office\_index].min\_weight &&

source->offices[source\_office\_index].packages[i].weight <= target->offices[target\_office\_index].max\_weight)

++n;

package\* newPackages = malloc(sizeof(package)\*(n + target->offices[target\_office\_index].packages\_count));

package\* oldPackages = malloc(sizeof(package)\*(source->offices[source\_office\_index].packages\_count - n));

for ( i = 0; i < target->offices[target\_office\_index].packages\_count; i++)

newPackages[i] = target->offices[target\_office\_index].packages[i];

n = target->offices[target\_office\_index].packages\_count;

int m = 0;

for ( i = 0; i < source->offices[source\_office\_index].packages\_count; i++)

if (source->offices[source\_office\_index].packages[i].weight >= target->offices[target\_office\_index].min\_weight &&

source->offices[source\_office\_index].packages[i].weight <= target->offices[target\_office\_index].max\_weight)

{

newPackages[n] = source->offices[source\_office\_index].packages[i];

++n;

}

else

{

oldPackages[m] = source->offices[source\_office\_index].packages[i];

++m;

}

target->offices[target\_office\_index].packages\_count = n;

free(target->offices[target\_office\_index].packages);

target->offices[target\_office\_index].packages = newPackages;

source->offices[source\_office\_index].packages\_count = m;

free(source->offices[source\_office\_index].packages);

source->offices[source\_office\_index].packages = oldPackages;

}

int number\_of\_packages(town t)

{

int ans = 0;

for ( i = 0; i < t.offices\_count; i++)

ans += t.offices[i].packages\_count;

return ans;

}

town town\_with\_most\_packages(town\* towns, int towns\_count)

{

int ans;

int max\_packages = -1;

for ( i = 0; i < towns\_count; i++)

if (number\_of\_packages(towns[i]) > max\_packages)

{

max\_packages = number\_of\_packages(towns[i]);

ans = i;

}

return towns[ans];

}

town\* find\_town(town\* towns, int towns\_count, char\* name)

{

for ( i = 0; i < towns\_count; i++)

if (!strcmp(towns[i].name, name))

return &(towns[i]);

return &towns[0];

}

int main()

{

int towns\_count;

scanf("%d", &towns\_count);

town\* towns = malloc(sizeof(town)\*towns\_count);

for ( i = 0; i < towns\_count; i++) {

towns[i].name = malloc(sizeof(char) \* MAX\_STRING\_LENGTH);

scanf("%s", towns[i].name);

scanf("%d", &towns[i].offices\_count);

towns[i].offices = malloc(sizeof(post\_office)\*towns[i].offices\_count);

for ( j = 0; j < towns[i].offices\_count; j++) {

scanf("%d%d%d", &towns[i].offices[j].packages\_count, &towns[i].offices[j].min\_weight, &towns[i].offices[j].max\_weight);

towns[i].offices[j].packages = malloc(sizeof(package)\*towns[i].offices[j].packages\_count);

for ( k = 0; k < towns[i].offices[j].packages\_count; k++) {

towns[i].offices[j].packages[k].id = malloc(sizeof(char) \* MAX\_STRING\_LENGTH);

scanf("%s", towns[i].offices[j].packages[k].id);

scanf("%d", &towns[i].offices[j].packages[k].weight);

}

}

}

int queries;

scanf("%d", &queries);

char town\_name[MAX\_STRING\_LENGTH];

while (queries--) {

int type;

scanf("%d", &type);

switch (type) {

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;

}