#include <stdio.h>

#include <stdlib.h>

#define MAX\_STRING\_LENGTH 6

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

{

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

for(int 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 max=target->offices[target\_office\_index].max\_weight;

int min=target->offices[target\_office\_index].min\_weight;

package\*temp=NULL;

int notaccept=0,weight;

post\_office\*s=source->offices+source\_office\_index;

post\_office\*d=target->offices+target\_office\_index;

for(int i=0;i<s->packages\_count;i++)

{

weight=s->packages[i].weight;

if(weight<=max&&weight>=min)

{

d->packages\_count++;

d->packages=(package\*)realloc(d->packages,sizeof(package)\*d->packages\_count);

d->packages[d->packages\_count- 1]=s->packages[i];

}

else

{

notaccept++;

temp=(package\*)realloc(temp,sizeof(package)\*notaccept);

temp[notaccept-1]=s->packages[i];

}

}

s->packages=temp;

s->packages\_count=notaccept;

}

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

town max\_town;

int sum,max=0;

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

{

sum=0;

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

sum+=towns[i].offices[j].packages\_count;

if(max<sum)

{

max\_town=towns[i];

max=sum;

}

}

return max\_town;

}

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

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

{

if(strcmp(towns[i].name,name)==0)

return towns +i;

}

return NULL;

}

int main()

{

    int towns\_count;

    scanf("%d", &towns\_count);

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

    for (int 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 (int 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 (int 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;

}