**Code**

#pragma warning (disable:4996)

#include <iostream>

#include <string>

#include <vector>

#include <utility>

#include <map>

#include <queue>

#define DEPART "Departure"

#define ARRIV "Arrival"

int tc, c, t, ti;

using namespace std;

typedef struct TABLE {

string name;

int start, end;

}Table;

typedef struct CITY {

string name;

vector<Table> times\_table;

}City;

int dest\_dp[100];

pair<int, int> back\_track[100];

string city, time;

string start, dest;

int start\_time, dest\_time;

map<string, int> cities;

vector<City> timeTable;

typedef struct \_NODE {

int start, end, idx;

}Node;

//pair<int, int> first is dest time, second is node index

struct myrule {

bool operator()(Node a, Node b) {

if (a.end == b.end) {

return a.start < b.start;

}

return a.end > b.end;

}

};

int string\_to\_int(string s) {

int t\_h = s[0] - '0';

int h = s[1] - '0';

int t\_m = s[2] - '0';

int m = s[3] - '0';

return (t\_h \* 10 + h) \* 60 + (t\_m \* 10 + m);

}

string int\_to\_string(int i) {

char t\_h = '0' + (i / 600);

i = i % 600;

char h = '0' + (i / 60);

i = i % 60;

char t\_m = '0' + (i / 10);

i = i % 10;

char m = '0' + i;

string s;

s.resize(4);

s[0] = t\_h;

s[1] = h;

s[2] = t\_m;

s[3] = m;

return s;

}

#define MAX\_NUM 0x1ffffff

void dijkstra(int idx1, int idx2) {

dest\_time = MAX\_NUM;

int num\_size = c;

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

dest\_dp[i] = MAX\_NUM;

}

priority\_queue<Node, vector<Node>, myrule> pq;

Node nd = { 0, start\_time, idx1 };

pq.push(nd);

while (pq.size() > 0 && num\_size > 0) {

int startTime = pq.top().start;

int destTime = pq.top().end;

int index = pq.top().idx;

pq.pop();

if (dest\_dp[index] != MAX\_NUM) continue;

dest\_dp[index] = destTime;

num\_size--;

for (auto tb : timeTable[index].times\_table) {

if (tb.start < destTime) continue;

int dest\_index = cities.find(tb.name)->second;

if (dest\_dp[dest\_index] > tb.end) {

Node nd1 = { tb.start, tb.end, dest\_index };

back\_track[dest\_index] = { index, tb.start };

pq.push(nd1);

}

}

}

}

int main() {

freopen("input.txt", "r", stdin);

freopen("output.txt", "w", stdout);

cin >> tc;

cerr << tc << endl;

for (int ts = 1; ts <= tc; ts++) {

cin >> c;

cerr << c << endl;

timeTable.resize(c);

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

cin >> city;

cerr << city << endl;

timeTable[i].name = city;

cities.insert(make\_pair(city, i));

}

cin >> t;

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

string time\_1, city\_1, time\_2, city\_2;

int idx1, idx2;

cin >> ti;

cerr << ti << endl;

cin >> time\_1 >> city\_1;

cerr << time\_1 << " " << city\_1 << endl;

int \_Time = string\_to\_int(time\_1);

for (int j = 1; j < ti; j++) {

idx1 = cities.find(city\_1)->second;

Table tb;

cin >> time\_2 >> city\_2;

cerr << time\_2 << " " << city\_2 << endl;

int \_Time2 = string\_to\_int(time\_2);

idx2 = cities.find(city\_2)->second;

tb = { city\_2, \_Time, \_Time2 };

timeTable[idx1].times\_table.push\_back(tb);

city\_1 = city\_2;

idx1 = idx2;

\_Time = \_Time2;

}

}

cin >> time >> start >> dest;

cerr << time << endl << start << endl << dest << endl << endl;

start\_time = string\_to\_int(time);

int idx1, idx2;

idx1 = cities.find(start)->second;

idx2 = cities.find(dest)->second;

dijkstra(idx1, idx2);

if (dest\_dp[idx2] != MAX\_NUM) {

int temp = idx2;

while (temp != idx1) {

if (back\_track[temp].first == idx1) break;

temp = back\_track[temp].first;

}

start\_time = back\_track[temp].second;

// show result

cerr.width(10);

cerr << DEPART;

cerr.width(10);

cerr << start;

cerr.width(10);

cerr << int\_to\_string(start\_time) << endl;

cerr.width(10);

cerr << ARRIV;

cerr.width(10);

cerr << start;

cerr.width(10);

cerr << int\_to\_string(dest\_dp[idx2]) << endl << endl;

cout << DEPART << " " << start << " " << int\_to\_string(start\_time) << endl;

cout << ARRIV << " " << dest << " " <<int\_to\_string(dest\_dp[idx2]) << endl;

}

else {

cout << "No connection" << endl;

cerr << "No connection" << endl;

}

//clear

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

back\_track[i].first = back\_track[i].second = 0;

}

cities.clear();

timeTable.clear();

}

return 0;

}

**Testcase**

3

3

Seoul

Suwon

Daejeon

3

2

0949 Seoul

1006 Suwon

2

1325 Seoul

1550 Daejeon

2

1205 Suwon

1411 Daejeon

0800

Seoul

Daejeon

3

Seoul

Suwon

Daejeon

3

2

0749 Seoul

1006 Suwon

2

1325 Seoul

1550 Daejeon

2

1205 Suwon

1411 Daejeon

0800

Seoul

Daejeon

3

Seoul

Suwon

Daejeon

3

2

0949 Seoul

1006 Suwon

2

1325 Seoul

1550 Daejeon

2

1205 Suwon

1411 Daejeon

1400

Seoul

Daejeon

**Output**

Departure Seoul 0949

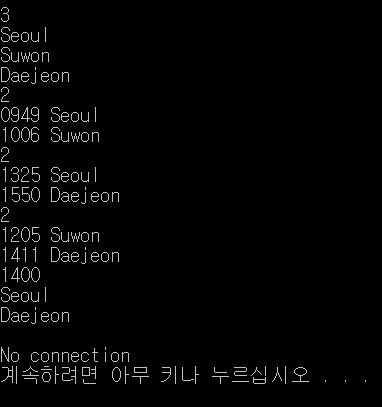
Arrival Daejeon 1411

Departure Seoul 1325

Arrival Daejeon 1550

No connection

**Screenshot**

****