题目描述：

智能手机方便了我们生活的同时，也侵占了我们不少的时间。“手机 App 防沉迷系统”能够让我们每天合理的规划手机 App 使用时间，在正确的时间做正确的事。

它的大概原理是这样的：

1、在一天 24 小时内，可注册每个 App 的允许使用时段；


2、一个时段只能使用一个App，举例说明：不能同时在09:00-10:00注册App2和App3；


3、App有优先级，数值越高，优先级越高。注册使用时段时，如果高优先级的App时间和低优先级的时段有冲突，则系统会自动注销低优先级的时段；如果App的优先级相同，则后添加的App不能注册。

举例1：

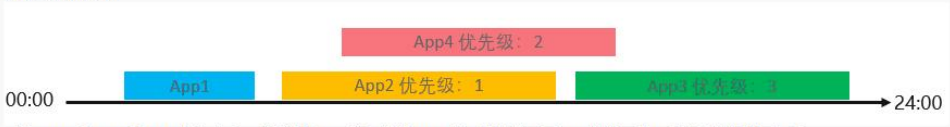
（1）注册App3前：


（2）App3注册时段和App2有冲突：


（3）App3优先级高，系统接受App3的注册，自动注销App2的注册：


举例2：

（1）注册App4：




（2）App4和App2及App3都有冲突，优先级比App2高，但比App3低，这种场景下App4注册不上，最终的注册效果如下：


4、一个App可以在一天内注册多个时段。


请编程实现，根据输入数据注册App，并根据输入的时间点，返回该时间点可用的App名称，如果该时间点没有注册任何App，请返回字符串"NA"。

输入描述：输入分3部分：第一行表示注册的App数N（N≤100）；第二部分包括N行，每行表示一条App注册数据；最后一行输入一个时间点，程序即返回该时间点的可用App。

2

App1 1 09:00 10:00

App2 2 11:00 11:30

09:30

数据说明如下：

1、N行注册数据以空格分隔，四项数据依次表示：App名称、优先级、起始时间、结束时间

2、优先级1-5，数字值越大，优先级越高

3、时间格式HH:MM，小时和分钟都是两位，不足两位前面补0

4、起始时间需小于结束时间，否则注册不上

5、注册信息中的时间段包含起始时间点，不包含结束时间点

输出描述：输出一个字符串，表示App名称，或NA表示空闲时间。

输入描述：

输入分 3 部分：第一行表示注册的 App 数 N（N≤100）；第二部分包括 N 行，每行表示一条 App 注册数据；最后一行输入一个时间点，程序即返回该时间点的可用 App。

2

App1 1 09:00 10:00

App2 2 11:00 11:30

09:30

数据说明如下：

1、N 行注册数据以空格分隔，四项数据依次表示：App 名称、优先级、起始时间、结束时间

2、优先级 1-5，数字值越大，优先级越高

3、时间格式 HH:MM，小时和分钟都是两位，不足两位前面补 0

4、起始时间需小于结束时间，否则注册不上

5、注册信息中的时间段包含起始时间点，不包含结束时间点

输出描述：

输出一个字符串，表示 App 名称，或 NA 表示空闲时间。

补充说明：

1、用例保证时间都介于 00:00-24:00 之间；
2、用例保证数据格式都是正确的，不用考虑数据输入行数不够、注册信息不完整、字符串非法、优先级超限、时间格式不正确的问题。

示例 1

输入：

```
1

App1 1 09:00 10:00

09:30
```

输出：

```
App1
```

说明：

App1 注册在 9 点到 10 点间，9 点半可用的应用名是 App1

示例 2

输入：

```
2

App1 1 09:00 10:00

App2 2 09:10 09:30

09:20
```

输出：

```
App2
```

说明：

App1 和 App2 的时段有冲突，App2 的优先级比 App1 高，注册 App2 后，系统将 App1 的注册信息自动注销后，09:20 时刻可用应用名是 App2.

示例 3
输入：

```
2
App1 1 09:00 10:00
App2 2 09:10 09:30
09:50
```

输出：

```
NA
```

说明：
App1 被注销后，09:50 时刻没有应用注册，因此输出 NA。

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

struct app {
    app(string _name, int _level, int _beg, int _end) : name(_name), level(_level),
        beg(_beg), end(_end) {}
    string name;
    int level;
    int beg;
    int end;
```

```
};
int main() {
    int n;
    string s;
    getline(cin, s);
    n = atoi(s.c_str());
    vector<app> vec;
    for (int i = 0; i < n; ++i) {
        string str;
        getline(cin, str);
        string name;
        int level;
        int k = 0;
        while (!isspace(str[k])) {
            name += str[k];
            k++;
        }
        while (isspace(str[k]))
            k++;
        string tmp;
        while (!isspace(str[k])) {
            tmp += str[k];
            k++;
        }
        level = atoi(tmp.c_str());
        while (isspace(str[k]))
            k++;
        tmp.clear();
        int a, b, c, d;
        sscanf(&str[k], "%d:%d %d:%d", &a, &b, &c, &d);
        vec.push_back(app(name, level, a * 60 + b, c * 60 + d));
    }
    auto comp = [](const app & app1, const app & app2) {
        return app1.level > app2.level;
    };
    sort(vec.begin(), vec.end(), comp);
    vector<bool> marked;
    marked.resize(vec.size());
    for (int i = 0; i < vec.size() - 1; ++i) {
        for (int j = i + 1; j < vec.size(); ++j) {
            if (marked[j])
                continue;
            else {
                if (vec[j].level < vec[i].level && ((vec[j].beg >= vec[i].beg &&
```

```
                                                        vec[j].beg < vec[i].end) ||
(vec[j].beg < vec[i].beg &&
                                                                vec[j].end       >
vec[i].beg)) )
                    marked[j] = true;
            }
        }
    }
    auto comp2 = [](const app & app1, const app & app2) {
        return app1.beg < app2.beg;
    };
    sort(vec.begin(), vec.end(), comp2);
    getline(cin, s);
    int a, b;
    sscanf(s.c_str(), "%d:%d", &a, &b);
    int time = a * 60 + b;
    int cur_level = -1;
    string output;
    for (int i = 0; i < vec.size(); ++i) {
        if (marked[i])
            continue;
        if ( time >= vec[i].beg && time < vec[i].end && vec[i].level >= cur_level)
{
            output = vec[i].name;
            cur_level = vec[i].level;
        }
    }
    if (cur_level == -1)
        cout << "NA";
    else
        cout << output;
    return 0;
}
```