

题目描述：

给定一个正整型数组表示待系统执行的任务列表，数组的每一个元素代表一个任务，元素的值表示该任务的类型。请计算执行完所有任务所需的最短时间。任务执行规则如下：

- 1、任务可以按任意顺序执行，且每个任务执行耗时间均为 1 个时间单位。
- 2、两个同类型的任务之间必须有长度为  $N$  个单位的冷却时间，比如： $N$  为 2 时，在时间  $K$  执行了类型 3 的任务，那么  $K+1$  和  $K+2$  两个时间不能执行类型 3 任务。
- 3、系统在任何一个单位时间内都可以执行一个任务，或者等待状态。

说明：数组最大长度为 1000,数组最大值 1000.

输入描述：

第一行记录一个用半角逗号分隔的数组，数组长度不超过 1000，数组元素的值不超过 1000

第二行记录任务冷却时间， $N$  为正整数， $N \leq 100$ 。

输出描述：

输出为执行完所有任务所需的最短时间。

补充说明：

```
示例1
输入：2,2,2,3
      2
输出：7
说明：时间1：执行类型2任务。
      时间2：执行类型3的任务（因为冷却时间为2，所以时间2不能执行类型2的任务）。
      时间3：系统等待（仍然在类型2的冷却时间）。
      时间4：执行类型2任务。
      时间5：系统等待。
      时间6：系统等待。
      时间7：执行类型2任务。
      因此总共耗时7。
```

```

1 #include <iostream>
2 #include <string>
3 #include <algorithm>
4 #include <unordered_map>
5 #include <vector>
6 #include <cstring>
7 using namespace std;
8
9 struct node{
10     string c;
11     int t,cold;
12     node(string cc, int tt, int cld):c(cc),t(tt), cold(cld){}
13     bool operator<(const node&a)const{
14         if(cold==a.cold){
15             return t>a.t;
16         }
17         return cold<a.cold;
18     }
19 };
20 unordered_map<string, int>mp;
21 int main() {
22     string ss;
23     int n,st=0,sum=0;
24     cin>>ss;
25     cin>>n;
26     ss+=" ";
27     mp.clear();
28     for(int i=0;i<ss.length();i++){
29         if(ss[i]!=' '){
30             string subs=ss.substr(st, i-st);
31             if(mp.find(subs)!=mp.end()){
32                 mp[subs]++;
33             }else{
34                 mp[subs]=1;
35             }
36             st=i+1;
37             sum++;
38         }
39     }
40     vector<node>v,tmp;
41     for(auto it:mp){
42         v.emplace_back(node{it.first,it.second,0});
43     }
44     int ans=0;
45     for(int i=0;i<sum;i++){
46         vector<node>tmp(v);
47         sort(tmp.begin(),tmp.end());
48         ans=tmp[0].cold+1;
49         v.clear();
50         if(tmp[0].t>1){
51             v.emplace_back(node{tmp[0].c,tmp[0].t-1,n});
52         }
53         for(int j=1;j<tmp.size();j++){
54             if(tmp[j].cold>0){
55                 tmp[j].cold=max(0, tmp[j].cold-tmp[0].cold-1);
56             }
57             v.emplace_back(tmp[j]);
58         }
59     }
60     cout<<ans<<endl;
61     return 0;
62 }
63 // 64 输出时请用 printf("lld")

```