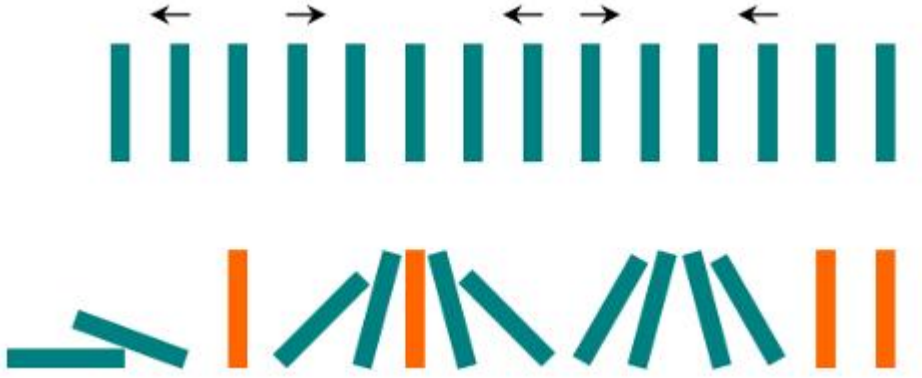
问题 A: 多米诺骨牌

小克里斯知道玩多米诺骨牌没有乐趣，他认为这太随机了，不需要技巧。相反，他决定玩多米诺骨牌，做一个“多米诺骨牌秀”。

克里斯将 n 张多米诺骨牌排成一条线，将每张多米诺骨牌垂直直立放置。一开始，他同时将一些多米诺骨牌推向左边或右边。，并且保证，在每两个多米诺骨牌推向相同方向之间的某个地方，至少有一个多米诺骨牌被推向相反的方向。

每一秒后，每张落在左边的多米诺骨牌都会把相邻的多米诺骨牌推到左边。同样，倒在右边的多米诺骨牌会推站在右边的相邻多米诺骨牌。当垂直多米诺骨牌的多米诺骨牌从两侧落在上面时，由于力量的平衡，它会保持静止。该图显示了该过程的一个可能示例。

给出克里斯推动多米诺骨牌的初始方向，找出在过程结束时垂直站立的多米诺骨牌数量！



输入第一行包含一个整数 n （1≤n≤3000），即行中多米诺骨牌的数量。下一行包含长度为 n 的字符串 s。字符串 si 的第 i 个字符等于

“L”，如果第 i 张多米诺骨牌被推到左侧;

“R”，如果第 i 张多米诺骨牌被推到右侧;

“.”，如果第 i 张多米诺骨牌没有被推。

可以保证，如果 si=sj=“L” 和 i<j，则存在这样的 k，即 i<k<j 和 sk=“R”;如果 si=sj=“R” 和 i<j，则存在这样的 k，即 i<k<j 和 sk=“L”。

输入	输出
14 .L.R...LR..L.. Output 4	5
5 R....	0
1 .	1

输出格式

输出单个整数，即在进程结束时保持垂直的多米诺骨牌的数量。

问题 B: 获奖人数

NOI2002 即将举行。为了增加观赏性，CCF 决定逐一评出每个选手的成绩，并直播即时的获奖分数线。本次竞赛的获奖率为 w%，即当前排名前 w% 的选手的最低成绩就是即时的分数线。
更具体地，若当前已评出了 p 个选手的成绩，则当前计划获奖人数为 max(1,⌊ p* w%⌋)其中 w 是获奖百分比，⌊ x⌋ 表示对 x 向下取整，max(x,y) 表示 x 和 y 中较大的数。如有选手成绩相同，则所有成绩并列的选手都能获奖，因此实际获奖人数可能比计划中多。
作为评测组的技术人员，请你帮 CCF 写一个直播程序。

输入格式
第一行有两个整数 n,w。分别代表选手总数与获奖率。
第二行有 n 个整数，依次代表逐一评出的选手成绩。
输出格式
只有一行，包含 n 个非负整数，依次代表选手成绩逐一评出后，即时的获奖分数线。相邻两个整数间用一个空格分隔。 $n<=10^5$

对于所有测试点，每个选手的成绩均为不超过 600 的非负整数，获奖百分比 w 是一个正整数且 1≤w≤99
提示:
在计算计划获奖人数时，如用浮点类型的变量（如 C/C++ 中的 float 、 double，Pascal 中的 real 、double 、extended 等）存储获奖比例 w%，则计算 5×60%时的结果可能为 3.000001，也可能为 2.999999，向下取整后的结果不确定。因此，建议仅使用整型变量，以计算出准确值。

10 60 200 300 400 500 600 600 0 300 200 100	200 300 400 400 400 500 400 400 300 300
10 30 100 100 600 100 100 100 100 100 100 100	100 100 600 600 600 600 100 100 100 100

样例 1 解释:

已评测选手人数	1	2	3	4	5	6	7	8	9	10
计划获奖人数	1	1	1	2	3	3	4	4	5	6
已评测选手的分数从高到低排列（其中，分数线用**粗体**标出）	**200**	**300** 200	**400** 300 200	500 **400** 300 200	600 500 **400** 300 200	600 600 **500** 400 300 200	600 600 500 **400** 300 200 0	600 600 500 **400** 300 300 200 0	600 600 500 400 **300** 300 200 200 0	600 600 500 400 300 **300** 200 200 100 0

注意，在第9名选手的成绩评出之后，计划获奖人数为5人，但由于有并列，实际会有6人获奖。

问题 C: 乘客的国籍

小 K 是一个海港的海关工作人员，每天都有许多船只到达海港，船上通常有很多来自不同国家的乘客。
小 K 对这些到达海港的船只非常感兴趣，他按照时间记录下了到达海港的每一艘船只情况；对于第 i 艘到达的船，他记录了这艘船到达的时间 ti (单位：秒)，船上的乘 客数 ki，以及每名乘客的国籍 x(i,1), x(i,2),…，x(i,k);。
小 K 统计了 n 艘船的信息，希望你帮忙计算出以每一艘船到达时间为止的 24 小时(24 小时=86400 秒）内所有乘船到达的乘客来自多少个不同的国家。
形式化地讲，你需要计算 n 条信息。对于输出的第 i 条信息，你需要统计满足 ti - 86400 < tp <= ti 的船只 p，在所有的 x(p,j)中，总共有多少个不同的数。

输入格式：
第一行输入一个正整数 n，表示小 K 统计了 n 艘船的信息。
接下来 n 行，每行描述一艘船的信息：前两个整数 ti 和 ki 分别表示这艘船到达海港的时间和船上的乘客数量，接下来 ki 个整数 x(i,j)表示船上乘客的国籍。
保证输入的 ti 是递增的，单位是秒；表示从小 K 第一次上班开始计时，这艘船在第 ti 秒到达海港。
$n<=10^5$， ki 的和$<=3*10^5$，$xij<=10^5$,$ti-1<=ti<=10^9$
输出：输出 n 行，第 i 行输出一个整数表示第 i 艘船到达后的统计信息。

输入	输出
3 1 4 4 1 2 2 2 2 2 3 10 1 3	3 4 4
4 1 4 1 2 2 3 3 2 2 3 86401 2 3 4 86402 1 5	3 3 3 4

【样例解释 1】
第一艘船在第 1 秒到达海港，最近 24 小时到达的船是第一艘船，共有 4 个乘客， 分别是来自国家 4,1,2,2，共来自 3 个不同的国家；
第二艘船在第 2 秒到达海港，最近 24 小时到达的船是第一艘船和第二艘船，共有 4 + 2 = 6 个乘客，分别是来自国家 4,1,2,2,2,3，共来自 4 个不同的国家；
第三艘船在第 10 秒到达海港，最近 24 小时到达的船是第一艘船、第二艘船和第 三艘船，共有 4+ 2+1=7 个乘客，分别是来自国家 4,1,2,2,2,3,3，共来自 4 个不同 的国家。

问题 D: 鸡冠

劳拉发现她在一间有宝藏的屋子里。令她惊讶的是，那里没有堆积成山的金子。环顾四周，她注意到在地面上有一张桌子，桌面拥有 n×m 个格子，每个格子上有一个数字。墙边有无数石头，桌边的柱子上有一条告示。告示上说，为了拿到宝藏，对桌子的每一行都必须选择从第一个格子开始连续的几个格子（不能为 0，至少要选一个格子）并且将石头放在上面，将这些格子压下去。那之后她会得到很多金币，数目等同于所有压下去的格子上的数字之和。

劳拉很快决定了如何放置石头，但在她开始之前她注意到了告示下面的一行小字。根据这行字，为了不让天花板掉下来并砸死探险者，探险者选择的格子要形成一个鸡冠形状。如果令 ci 表示第 i 行选择的格子数量，那么选择的格子能形成一个鸡冠，当且仅当 c1>c2<c3>c4…，就是相邻的不等号方向不同。现在劳拉很迷惑，停止了思考，不知道要怎么做。帮她判断她最多能获得多少金币，同时活下来。

输入格式

第一行包含一对整数 n，m（2≤n，m≤1500). 接下来的 n 行每行包含 m 个整数—桌子本身。表中数字的绝对值不超过 10000。

输出格式

打印劳拉可以获得的最大硬币数量。

2 2 -1 2 1 3	2

Examples

Input

Output

2

问题 E:奶牛线路

厌倦了农场寒冷的冬季天气，奶牛贝茜计划飞往一个温暖的目的地度假。

不幸的是，她发现只有一家航空公司，博维尼亚航空，愿意向奶牛出售机票，而且这些机票的结构有些复杂。

博维尼亚航空公司拥有 N 架飞机，每架飞机都在由两个或多个城市组成的特定“航线”上飞行。例如，一架飞机可能从城市 1 起飞，然后飞到城市 5，然后飞到城市 2，最后飞到城市 8。没有一个城市会在一条航线上出现多次。

如果贝茜选择了一条航线，那么她可以在航线上的任何城市登机，然后在航线上的任何城市下飞机。

她不需要在航线的第一个城市登机或在最后一个城市下机。

每条航线都有一定的费用，只要贝茜乘坐了某个航线，不论乘坐时途径的城市有多少，都需要支付全部的航线费用。

如果贝茜在旅途中多次搭乘某个航线（也就是说，如果她离开了该航线，后来又从另一个城市回来乘坐该航线），那么每次乘坐该航线时，她都必须为此付费。贝西想找到从她所在的农场（在 A 市）到热带目的地（B 市）的最便宜的旅行方式。

请帮助她确定她需要支付的最低费用是多少，以及为达到该最低费用所需的最少搭乘航班次数。

输入第一行包含三个整数 A,B,N。接下来 2N 行，每两行描述一条航线，第一行包含航线的乘坐费用以及航线途径的城市数量。第二行包含按航线顺序排列的城市列表。

输出在一行输出贝茜从 A 到 B 所需要支付的最低费用以及所需的最少搭乘航班次数。

如果无法到达目的地，则输出 -1 -1。

数据范围

$1 \le N \le 1000$,

$1 \le$ 航线费用 $\le 10^9$,

$1 \le$ 航线途径城市数量 ≤ 100,

城市编号范围 [1,1000]。

3 4 3 3 5 1 2 3 4 5 2 3 3 5 4 1 2 1 5	2 2

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

using namespace std;

int f(const string &s, const string &t)
{
    int n = s.length(), m = t.length();

    vector<int> shift(128, m + 1);

    int i, j;

    for (j = 0; j < m; j++)
        shift[t[j]] = m - j;

    for (i =0; i<= n - m; i += shift[s[i + m]]){
        j =0;
        while(j < m && s[i +j] == t[j]) j++;
        if (j == m) return i;
    }

    return -1;
}

int main()
{
    string a ,b;
    cin >> a >> b;
    cout << f(a, b) << endl;
    return 0;
}
```

假设输入字符串由 ASCII 可见字符组成，完成下面的判断题和单选题：

判断题

1. （1 分）当输入为“abcde fg”时，输出为-1。
2. 当输入为“abbababbbab abab”时，输出为 4。
3. 当输入为“GoodLuckCsp2022 22”时，第 20 行的“j++”语句执行次数为 2。

单选题

4. 该算法最坏情况下的时间复杂度为（ ）。
5. **f(a, b)** 与下列（ ）语句的功能最**类似**。
6. 当输入为“baaabaaabaaabaaaa aaaa”，第 20 行的“j++”语句执行次数为 （ ）。

1. A. 正确
   B. 错误

2. A. 正确
   B. 错误

3. A. 正确
   B. 错误

4. A. $O(n+m)$
   B. $O(nlogm)$
   C. $O(mlogn)$
   D. $O(nm)$

5. A. a.find(b)
   B. a.rfind(b)
   C. a.substr(b)
   D. a.compare(b)

6. A. 9
   B. 10
   C. 11
   D. 12

```cpp
#include <iostream>

using namespace std;

const int MAXN = 105;

int n, m, k, val[MAXN];
int temp[MAXN], cnt[MAXN];

void init()
{
    cin >> n >> k;
    for (int i = 0; i < n; i++) cin >> val[i];
    int maximum = val[0];
    for (int i = 1; i < n; i++)
        if (val[i] > maximum) maximum = val[i];
    m = 1;
    while (maximum >= k) {
        maximum /= k;
        m++;
    }
}

void solve()
{
    int base = 1;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < k; j++) cnt[j] = 0;
        for (int j = 0; j < n; j++) cnt[val[j] / base % k]++;
        for (int j = 1; j < k; j++) cnt[j] += cnt[j - 1];
        for (int j = n - 1; j >= 0; j--) {
            temp[cnt[val[j] / base % k] - 1] = val[j];
            cnt[val[j] / base % k]--;
        }
        for (int j = 0; j < n; j++) val[j] = temp[j];
        base *= k;
    }
}

int main()
{
    init();
    solve();
    for (int i = 0; i < n; i++) cout << val[i] << ;
    cout << endl;
    return 0;
}
```

假设输入的 n 为不大于 100 的正整数，k 为不小于 2 且不大于 100 的正整数，val[i]在 int 表示范围内，完成下面的判断题和单选题：

判断题

1. 这是一个不稳定的排序算法。（ ）
2. 该算法的空间复杂度仅与 n 有关。（ ）
3. 该算法的时间复杂度为$O(m(n+k))$。（ ）

单选题

4. 当输入为“5 3 98 26 91 37 46”时，程序第一次执行到第 36 行，val[]数组的内容依次为（ ）。
5. 若 val[i]的最大值为 100，k 取（ ）时算法运算次数最少。
6. 当输入的 k 比 val[i]的最大值还大时，该算法退化为（ ）算法。

1. ○ A. 正确
   ○ B. 错误

2. ○ A. 正确
   ○ B. 错误

3. ○ A. 正确
   ○ B. 错误

4. ○ A. 91 26 46 37 98
   ○ B. 91 46 37 26 98
   ○ C. 98 26 46 91 37
   ○ D. 91 37 46 98 26

5. ○ A. 2
   ○ B. 3
   ○ C. 10
   ○ D. 不确定

6. ○ A. 选择排序
   ○ B. 冒泡排序
   ○ C. 计数排序
   ○ D. 桶排序