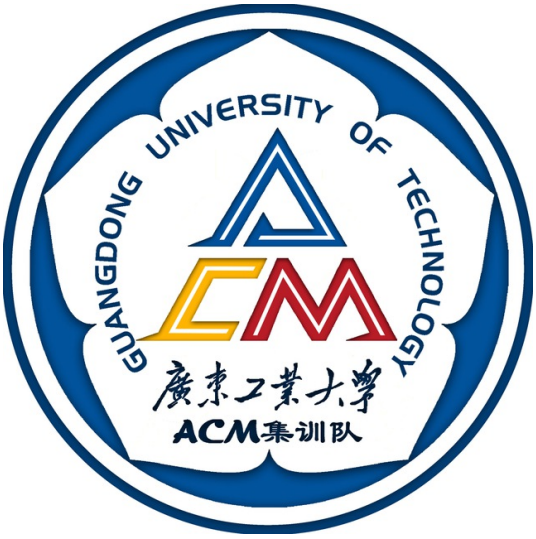
2025 年广东工业大学 ACM 程序设计竞赛月赛

广东工业大学 ACM 集训队

2025 年 10 月 19 日



试题列表

A	%%%自动机
B	k-冲突数对
C	序列重构（简单版）
D	序列重构（困难版）
E	新田忌赛马
F	雪莉的预言
G	俄罗斯方块的博弈
H	我不吃水果
I	运动世界校园
J	逃出生天
K	最不上升也不下降序列
L	翻转硬币
M	字符消消乐
N	缩花

共 14 题

Problem A. %%%自动机

输入文件: standard input
输出文件: standard output

5 月 11 日，第十届中国大学生程序设计竞赛全国总决赛（简称 CCPC Final）在香港科技大学（广州）圆满落幕。广东工业大学派出"打 ACM 导致的"参赛队伍，以学校排名第四、队伍排名第五成绩摘得金奖，紧随北京大学、上海交通大学、清华大学之后，超越诸多双一流传统强校，创造地方高校在该赛事中的历史性突破。

强校环伺中突围广工学子展现硬核实力

本届 CCPC Final 汇聚了全国 107 高校的 122 支精英队伍，包括北京大学、清华大学、上海交通大学、北京交通大学等连续多年称霸赛场的"王牌战队"。广工"打 ACM 导致的"队伍凭借对算法的深刻理解与高效协作能力，在 5 小时极限编程中连续攻克 7 道高难度赛题，以解题数、罚时双优表现位列高校排名第四、队伍排名第五。

随着全国高校对 ACM 竞赛的日益重视，比赛竞争也越来越激烈，强校对 ACM 竞赛的垄断日益加剧。本届 CCPC Final，全国前十名中的九席被"双一流"高校占据，广东工业大学是唯一以地方院校身份入围的队伍，生源和名校存在较大差距，取得这样的成绩殊为不易。

学科竞赛体系筑基人才培养模式成效凸显

广工此次突破并非偶然。自 2002 年组建 ACM 集训队以来，成功构建了"阶梯式培养 + 产教融合"的特色模式。日常训练覆盖数据结构、算法设计等方面，年均训练时长超 1500 小时。计算机学院对 ACM 集训队从训练场地、竞赛经费、招生政策等给与了大力支持。同时，学院与华为、腾讯、字节以及文远知行等著名企业建立合作关系，每年 ACM 队均有近 20 名同学进入头部企业或选择继续深造。

近两年，广工 ACM 队已展现强劲势头。在 2024 年 ICPC 亚洲赛以及 CCPC 分站赛中，累计获得 ICPC 亚洲赛金奖 3 枚，CCPC 金奖 1 枚。在 5 月 4 号举行的 2025 年 ICPC 国际大学生程序设计竞赛全国邀请赛（陕西）站比赛中，"打 ACM 导致的"队获得亚军。

竞赛精神薪火相传创新基因融入发展血脉

广工 ACM 集训队自成立以来，累计三次打进 ICPC 国际大学生程序设计竞赛世界总决赛，获 ICPC 国际大学生程序设计竞赛亚洲赛金奖 22 枚，CCPC 中国大学生程序设计竞赛 10 枚（包括 CCPC Final 金奖 3 枚）。

随着"冲一流"建设的不断推进，广工正以算法竞赛为支点，推动计算机学科与人工智能、工业软件等领域的深度融合，为粤港澳大湾区数字经济高质量发展培养更多"硬核"创新人才。

有 n 位同学对此表示膜拜，请你帮他们输出一个字符串，由 n 个字符 '%' 组成。

输入格式

一行一个整数 n（$1 \leq n \leq 100$）。

输出格式

一行一个字符串，由 n 个字符 '%' 组成。

样例

standard input	standard output
3	%%%

Problem B. k-冲突数对

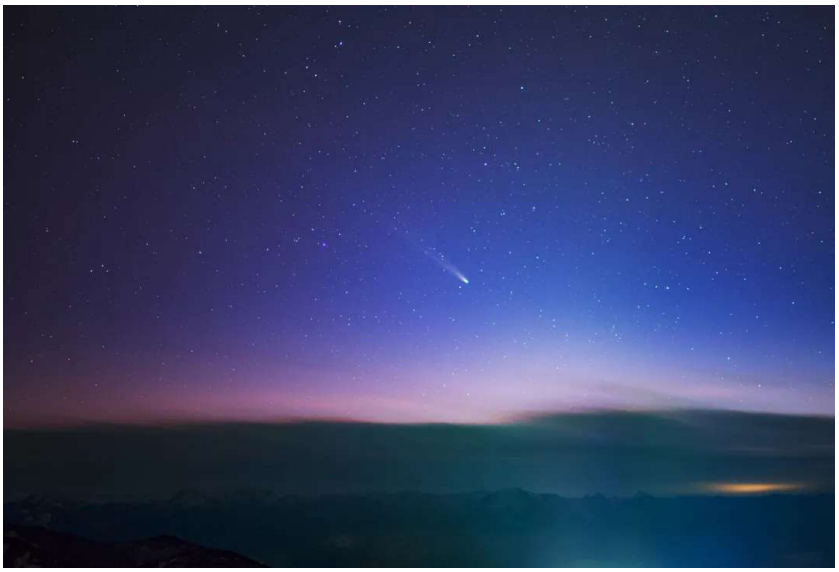
输入文件: standard input
输出文件: standard output

我们遥望却不在同一片光年下燃烧
如同赫利孔山的清泉与尘世的井
数字在深潭底静默保持各自的晶莹

没有两颗星会被相同的引力弯曲轨道
当猎户座腰带亮起天狼便退入寂寥
这是银河的契约：永不相交的辉芒

所有试图靠近的轨迹都将坠入虚无
像被风驱散的种子各自寻找孤独的土壤
唯有保持距离的星座才能通过时间的长廊

秋夜如此辽阔足够容纳每盏独立的灯
天鹅飞越天河时翅膀从不触碰波浪
正如我们怀抱完整的光在黑暗中平行穿行



请注意本题特殊的时空限制，并使用较快的读入方式。

定义 “k-冲突数对” 为满足 $x + y - \gcd(x, y) = k$ 的数对 (x, y) ，其中 $\gcd(x, y)$ 表示 x 和 y 的最大公因数。

现在给定一个长度为 n 的数组 $[a_1, a_2, \ldots, a_n]$ ，**保证所有 a_i 两两不同**。

对于 $1 \le l \le r \le n$ ，如果 **不存在** 两个（可能相同的）下标 $l \le i, j \le r$ ，满足 (a_i, a_j) 为 “k-冲突数对” ，则称子数组 $[a_l, a_{l+1}, \ldots, a_r]$ 是合法的。特殊地，空数组 [] 也是 a 的子数组。

你需要求出数组 a 的 **最长** 合法子数组的长度。

输入格式

第一行 2 个整数 n 和 k（$1 \leq n, k \leq 1081080$）。

第二行 n 个整数 $a_1, a_2, \ldots, a_n$（$1 \leq a_i \leq 1081080^2$，且所有 a_i 两两不同）。

输出格式

输出一个整数，表示最长合法子数组的长度。

样例

standard input	standard output
1 1 1	0
4 4 2 3 4 5	1
7 8 4 3 6 7 2 5 1	3
5 114514 11 45 14 1919 810	5

提示说明

在第一个样例中，(a_1, a_1) 为冲突数对，最长合法连续子数组为 [] ，长度为 0 。

在第二个样例中，最长合法连续子数组为 [2] 、[3] 和 [5] ，长度为 1 。

在第三个样例中，最长合法连续子数组为 $[3, 6, 7]$ 和 $[2, 5, 1]$ ，长度为 3 。

Problem C. 序列重构（简单版）

输入文件: standard input
输出文件: standard output

本题是一道交互题，选手的程序可以向交互器进行若干询问，最后进行回答。具体交互方式请参考下方交互协议部分。

这是该问题的简单版本。不同版本之间的区别在于，在此版本中，你可以查询长度为 n 的任意序列。

定义：一个长度为 n 的排列是指由 1 到 n 的所有整数构成，每个数 **恰好出现一次** 的序列。

现在有一个长度为 n 的未知排列 p 。

为了确定排列 p ，你可以进行最多 n^2 次查询，每次查询方式如下：

- 输出一个长度为 n 的 **任意序列** a。
- 交互器会计算 a 与 p 在同一位置上数值相同的个数 k（即满足 $a_i = p_i$ 的 i 的数量），并输出该 k 值。

在完成最多 n^2 次查询后，你需要输出排列 p 的全部元素。

请注意，排列 p 是在任何查询之前就固定的，且不依赖任何查询。

输入格式

每个测试包含多个测试用例。

第一行输入一个整数 t（$1 \le t \le 2500$），表示测试用例数量。

每个测试用例包括一行一个整数 n（$1 \le n \le 100$），表示排列的长度。

保证所有测试用例的 n^2 之和不超过 10000。

交互协议

每个测试用例的程序交互应从读取整数 n 开始。

要进行查询，请按以下格式输出一行 n 个整数，表示你提供的 **序列** a ：

- ? a_1 a_2 ... a_n

注意，你需要保证你输出的每个 a_i 均不超过 32 位有符号整数可表示的范围。

然后你会收到一个整数 k（$0 \le k \le n$），表示 a 和 p 有多少位置是相同的。

要报告最终答案，请按以下格式输出一行：

- ! p_1 p_2 ... p_n

请注意，回答问题不计入查询次数限制。

在输出答案后，您的程序应继续处理下一个测试用例，若无更多测试用例则终止程序。

此外，在每次输出完毕后，你都需要立即 **刷新缓冲区！** 你可以使用如下语句来刷新缓冲区：

- 对于 C/C++：`fflush(stdout)`
- 对于 C++：`std::cout << std::flush`
- 对于 Java：`System.out.flush()`
- 对于 Python：`stdout.flush()`

对于 C++ 语言，在输出换行时如果你使用 `std::endl` 而不是 '\n'，也可以自动刷新缓冲区。

样例

standard input	standard output
2 1 3 0 1 2 3	 ! 1 ? -11 45 14 ? 1 2 3 ? 1 1 3 ? 2 1 3 ! 2 1 3

提示说明

请注意，示例交互仅用于理解题意，并不保证能找到唯一排列 p 。

Problem D. 序列重构（困难版）

输入文件: standard input
输出文件: standard output

本题是一道交互题，选手的程序可以向交互器进行若干询问，最后进行回答。具体交互方式请参考下方交互协议部分。

这是该问题的困难版本。不同版本之间的区别在于，在此版本中，你只能查询长度为 n 的排列。

定义：一个长度为 n 的排列是指由 1 到 n 的所有整数构成，每个数 **恰好出现一次** 的序列。

现在有一个长度为 n 的未知排列 p 。

为了确定排列 p ，你可以进行最多 n^2 次查询，每次查询方式如下：

- 输出一个长度为 n 的 **排列** a。
- 交互器会计算 a 与 p 在同一位置上数值相同的个数 k（即满足 $a_i = p_i$ 的 i 的数量），并输出该 k 值。

在完成最多 n^2 次查询后，你需要输出排列 p 的全部元素。

请注意，排列 p 是在任何查询之前就固定的，且不依赖任何查询。

输入格式

每个测试包含多个测试用例。

第一行输入一个整数 t（$1 \le t \le 2500$），表示测试用例数量。

每个测试用例包括一行一个整数 n（$1 \le n \le 100$），表示排列的长度。

保证所有测试用例的 n^2 之和不超过 10000。

交互协议

每个测试用例的程序交互应从读取整数 n 开始。

要进行查询，请按以下格式输出一行 n 个整数，表示你提供的 **排列** a ：

- ? a_1 a_2 ... a_n

注意，你需要保证你输出的序列 a 满足由 1 到 n 的所有整数构成，每个数恰好出现一次。

然后你会收到一个整数 k（$0 \le k \le n$），表示 a 和 p 有多少位置是相同的。

要报告最终答案，请按以下格式输出一行：

- ! p_1 p_2 ... p_n

请注意，回答问题不计入查询次数限制。

在输出答案后，您的程序应继续处理下一个测试用例，若无更多测试用例则终止程序。

此外，在每次输出完毕后，你都需要立即 **刷新缓冲区！** 你可以使用如下语句来刷新缓冲区：

- 对于 C/C++：`fflush(stdout)`
- 对于 C++：`std::cout << std::flush`
- 对于 Java：`System.out.flush()`
- 对于 Python：`stdout.flush()`

对于 C++ 语言，在输出换行时如果你使用 `std::endl` 而不是 '`\n`'，也可以自动刷新缓冲区。

样例

standard input	standard output
2 1 3 1 0 3 	 ! 1 ? 1 2 3 ? 1 3 2 ? 2 1 3 ! 2 1 3

提示说明

请注意，示例交互仅用于理解题意，并不保证能找到唯一排列 p 。

Problem E. 新田忌赛马

输入文件: standard input
输出文件: standard output

本题数据量较大，可能需要使用较快的读入方式。

田忌与齐王再次进行赛马比赛。这次比赛规则有所不同，田忌需要通过合理安排马匹的对阵顺序，最大化自己的赏金收益。

田忌有 n 匹马，第 i 匹马的速度为 a_i，齐王有 m 匹马，第 i 匹马的速度为 b_i。由于田忌对自己和齐王的马匹了如指掌，他知道他和齐王的马都是按速度降序排列的。

每次比赛，田忌可以选择自己的一匹从未出战的马 i 与齐王的一匹从未出战的马 j 进行比赛：

1. 如果田忌的马速度严格大于齐王的马（$a_i > b_j$），田忌将获得 b_j 的赏金。
2. 如果田忌的马速度小于或等于齐王的马（$a_i \le b_j$），田忌不会获得赏金。

你需要计算田忌能够获得的 **最大** 赏金总额。

输入格式

第一行输入两个整数 n 和 m（$1 \le n, m \le 5 \times 10^5$），分别表示田忌和齐王的马匹数量。

第二行输入 n 个整数 $a_1, a_2, \ldots, a_n$（$1 \le a_i \le 10^9$），表示田忌各匹马的速度。

第三行输入 m 个整数 $b_1, b_2, \ldots, b_m$（$1 \le b_i \le 10^9$），表示齐王各匹马的速度。

保证对 $i \in [1, n-1]$，有 $a_i \ge a_{i+1}$；对 $i \in [1, m-1]$，有 $b_i \ge b_{i+1}$。

输出格式

输出一个整数，表示田忌能够获得的最大赏金总额。

样例

standard input	standard output
2 2 3 1 4 2	2
3 3 11 10 7 10 9 8	19
6 7 6 5 4 3 2 1 7 6 5 4 3 2 1	15

Problem F. 雪莉的预言

输入文件: standard input
输出文件: standard output

审判的钟声敲响，m 名少女齐聚审判庭——今日，一桩命案震惊众人。

她们必须在投票中找出真凶。若失败，所有人将被当作魔女，一同处刑。

雪莉获得了一份神秘的预言：

“在处刑投票过程中，频繁处于票数第 k 大位置的少女，极有可能就是凶手。”

每当有人新增投票或撤回投票后，雪莉会将所有少女的得票数按 **降序** 记成一个长度为 m 的数组 a。此时，**所有** 得票数等于 a_k 的少女，都被视为“票数第 k 大”，并被记录下来。

在雪莉开始记录前，审判庭已经有了 n 张投票。然而，审判庭秩序骤然崩坏——接连发生了 q 次来历不明的投票或撤票操作。面对海量数据，雪莉难以及时处理，于是向你求助。

少女们从 0 到 $m-1$ 编号。为简化任务，你只需在每次操作后，计算所有“票数第 k 大”的少女的编号之和。设第 i 次操作后的和为 S_i，为减少输出量，请你最终输出一个整数：$S_1 \oplus S_2 \oplus \cdots \oplus S_q$，其中 $\oplus$ 表示按位异或运算。

输入格式

第一行包含六个整数 $seed$, k, n, m, q, b，其中：

- $seed$（$1 \le seed \le 10^9$）：随机数生成器的种子
- k（$1 \le k \le 2 \times 10^7$）：需要查询的出现次数第 k 大的数
- n（$1 \le n \le 2 \times 10^7$）：审判庭已有的票数
- m（$1 \le m \le 2 \times 10^7$）：少女们的人数

- q（$1 \le q \le 2 \times 10^7$）：投票操作的数量
- b（$0 \le b \le 10^9 + 7$）：控制操作类型的参数

特别地，保证 $n + q \le 2 \times 10^7$ 。

由于输入数据量非常大，你需要使用指定的随机数生成器生成其余数据：

```
long long seed;
long long read() {
    seed = (seed * 13331 + 23333) % 1000000007;
    return seed;
}
```

初始的 n 次投票通过 n 次随机过程生成，每个投票所指的少女编号为 `read() % m` 。

接下来的 q 个操作也通过随机过程生成。对于每个操作：

1. 生成随机数 `read()` ，记为 op
2. 如果 $op \ge b$，则执行投票操作，投票所指的少女编号为 `read() % m`
3. 如果 $op < b$，则执行撤票操作，撤票所指的少女编号为 `read() % m`

注意：如果少女本身没有票，则 **忽略** 对少女的撤票操作，但你 **仍需要计算一次答案** 。

输出格式

设第 i 次操作后目标少女们的编号和为 S_i，为减少输出量，请你最终输出一个整数：$S_1 \oplus S_2 \oplus \cdots \oplus S_q$，其中 $\oplus$ 表示按位异或运算。

样例

standard input	standard output
114514 2 5 5 5 500000000	8
791372847 1 5 5 5 0	2

提示说明

在第一个样例中：

初始投票为 $[0, 3, 1, 1, 3]$，少女 0 到 4 的票数分别为 $[1, 2, 0, 2, 0]$。排序后数组为 $[2, 2, 1, 0, 0]$，$a_k = a_2 = 2$，则第 2 大的票数为 2，而少女 1 和 3 的票数均为 2，因此它们都是“票数第 2 大的少女”。

- 第一次执行投票 4 号操作，票型变为 $[1,2,0,2,1]$，1 和 3 是票数第 2 大的少女，结果为 $1+3=4$ 。
- 第二次执行撤票 0 号操作，票型变为 $[0,2,0,2,1]$，结果为 $1+3=4$ 。
- 第三次执行投票 4 号操作，票型变为 $[0,2,0,2,2]$，结果为 $1+3+4=8$ 。
- 第四次执行撤票 3 号操作，票型变为 $[0,2,0,1,2]$，结果为 $1+4=5$ 。
- 第五次执行撤票 0 号操作，票型保持不变，结果为 $1+4=5$ 。

所有操作结果的异或值为 $4 \oplus 4 \oplus 8 \oplus 5 \oplus 5 = 8$ 。

Problem G. 俄罗斯方块的博弈

输入文件: standard input
输出文件: standard output

Alice 和 Bob 喜欢玩一种方块游戏。他们有一个 4×4 的网格棋盘，以及两种类型的方块：

- x 个 T 形方块（$0 \le x \le 3$）
- y 个 L 形方块（$0 \le y \le 3$）

T 形方块可以旋转，平移和翻转，覆盖 3 个相邻的格子和正中间延伸出的第 4 个格子（如图，类似俄罗斯方块中的 T 方块）。

L 形方块也可以旋转，平移和翻转，覆盖 3 个相邻的格子和拐角处延伸出的第 4 个格子（如图，类似俄罗斯方块中的 L 方块）。

Alice 先手，两人轮流在棋盘上放置方块。每个方块必须完整地放置在棋盘内，且 **不能与棋盘中已有的其它方块重叠** 。无法放置方块的玩家输掉游戏。

假设 Alice 和 Bob 都足够聪明，他们会选择对自己最优的策略进行游戏。

现在给定 x 和 y，请判断 Alice 是否有必胜策略。

输入格式

第一行输入两个整数 x 和 y（$0 \le x, y \le 3$），表示 T 形方块和 L 形方块的数量。

输出格式

如果 Alice 有必胜策略，输出一行 “Alice” ，否则，输出一行 “Bob” 。

样例

standard input	standard output
0 0	Bob
1 0	Alice
3 3	Alice

Problem H. 我不吃水果

输入文件: standard input
输出文件: standard output

有一个大小为 $n \times n$ 的正方形水果蛋糕，我们将其看作由 n 行 n 列共 $n \times n$ 个大小均为 1 的位置组成，其中某些位置上放有水果。

这个蛋糕的分割方式比较特殊，只能在行与行的分界处、列与列的分界处分割。

小 Z 不吃水果，他想从左上角切一块最大的没有水果的矩形蛋糕。换句话说，他需要找到最大的矩形区域，包含第一行第一列，其中的所有位置都没有水果。

请你帮忙计算小 Z 能切到的 **最大** 蛋糕大小。

输入格式

第一行 1 个整数 n（$1 \leq n \leq 1000$）。

接下来 n 行每行 n 个整数，对第 i 行第 j 列的整数：

- 为 0 表示第 i 行第 j 列没有水果。
- 为 1 表示第 i 行第 j 列放有水果。

输出格式

一行一个整数，表示小 Z 能切到的最大蛋糕大小。

样例

standard input	standard output
1 0	1
3 0 0 0 0 0 1 1 1 1	4
3 1 0 0 1 0 1 1 1 1	0

Problem I. 运动世界校园

输入文件: standard input
输出文件: standard output

总所周知，运动世界校园是一个广受好评的校园跑软件，凡是用过它的同学都对它赞不绝口。

现在小 L 正打算开始校园跑，在他开始跑步之前，需要获取点位。

由于运动世界校园是一个制作精良的校园跑软件，所以每次尝试获取点位足足有 p 概率获取成功!

小 L 有三次尝试获取点位的机会，每次点位获取是相互独立的，只要任意一次获取点位成功，那么就视为点位获取成功；否则在接下来的一段时间内，他将无法再获取点位（那一定是小 L 自己的问题）。

请你帮小 L 计算他在三次尝试内点位获取成功的概率。

输入格式

一行一个浮点数 p（$0 \le p \le 1$）。

输出格式

输出一个浮点数，表示三次尝试内点位获取成功的概率。

如果你的答案与正确答案的绝对误差或相对误差不超过 10^{-4}，你的答案将被视为正确。

形式化地说，设你的答案为 a，正确答案为 b。当且仅当满足 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-4}$ 时，你的答案才会被视为正确。

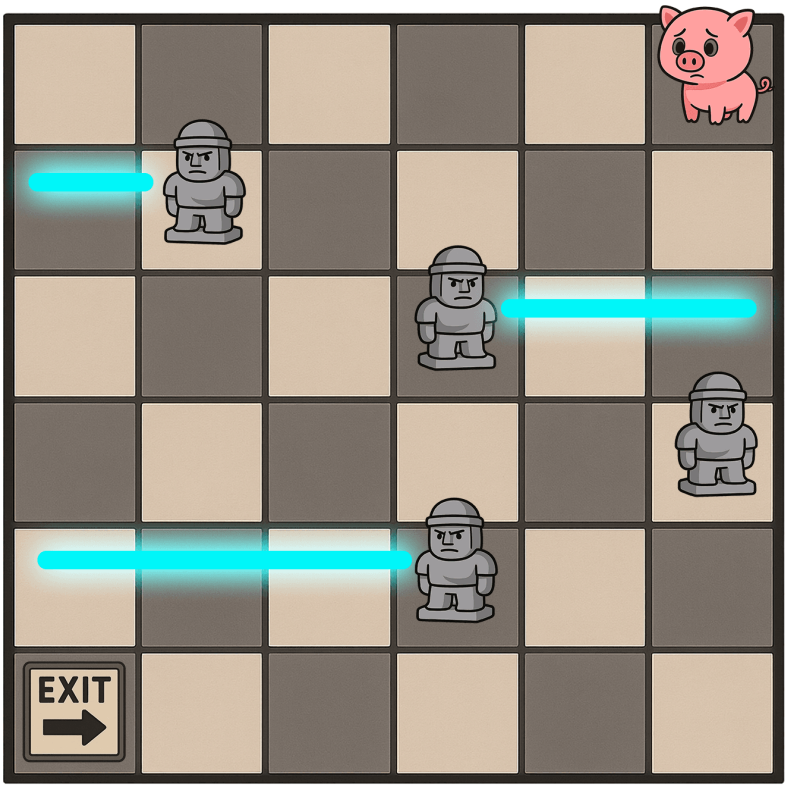
样例

standard input	standard output
0.5	0.875000

Problem J. 逃出生天

输入文件: standard input
输出文件: standard output

在一个 $n+2$ 行 m 列的网格世界中，一只可怜的小猪被困在了右上角的位置 $(1,m)$。它的目标是到达左下角的出口 $(n+2,1)$，但网格中充满了危险，除了第 1 行和第 $n+2$ 行，每一行都有一只石像在发射激光！



如图所示，所有石像有且仅有左和右两种朝向，每只石像会朝其正前方发射一道激光，被激光照射到的区域均无法通行。

每时刻，小猪可以选择移动到相邻格子。也就是说，如果小猪在当前时刻前处在 (x,y) ，则下一时刻小猪可以移动到网格范围内的 $(x-1,y),(x+1,y),(x,y-1),(x,y+1)$ 其中之一。

在 **任意时刻** ，小猪不能移动到网格之外，不能与石像或处在同一格子，也不能进入被激光照射到的区域中。

请判断小猪是否可以达成目标。

输入格式

第一行包含两个整数 n,m（$1\le n,m\le 1000$），含义如题面所示。

接下来 n 行，第 i 行包含一个整数 c_i（$1\le c_i\le m$）和一个字符 d_i（$d_i\in\{L,R\}$）。

- c_i 表示第 $i+1$ 行石像的初始列位置。
- d_i 表示第 $i+1$ 行石像的方向，'L' 表示向左，'R' 表示向右。

输出格式

输出一行字符串，如果小猪能够到达出口，输出 “YES”，否则输出 “NO” 。

你可以以任意大小写输出 “YES” 和 “NO”（例如，字符串 “yEs”、“yes”、“Yes” 和 “YES” 将被识别为肯定的回答）。

样例

standard input	standard output
2 2 2 R 1 R	NO
4 6 2 L 4 R 6 R 4 L	YES

Problem K. 最不上升也不下降序列

输入文件: standard input
输出文件: standard output

在创世的第七日，上帝开始创造序列世界。

“要有排列”

于是，排列诞生了—— 1 到 n 的每个整数恰好出现一次的序列。

“要有子序列”

于是，子序列诞生了——从序列中删除零个或多个元素，保持剩余元素顺序，得到的新序列。

“要有上升”

于是，最长上升子序列（LIS）诞生了——最长的严格递增的子序列。

“要有下降”

于是，最长下降子序列（LDS）诞生了——最长的严格递减的子序列。

上帝看着这一切，沉思片刻，说：

“要既不上升也不下降”

作为序列世界的造物主，你需要构造一个长度为 n 的排列，让 LIS 长度与 LDS 长度之和 **最小** 。

输入格式

输入一个整数 n（$1 \le n \le 2000$），表示排列的长度。

输出格式

输出一行 n 个整数，表示你构造的排列，其 LIS 长度与 LDS 长度之和最小。

如果有多种可能的排列，输出任意一种即可。

样例

standard input	standard output
1	1
3	1 3 2
7	1 4 3 2 7 6 5

Problem L. 翻转硬币

输入文件: standard input
输出文件: standard output

小 L 在玩一个翻转硬币的游戏，其规则如下：

有 n 个硬币按顺序排成一行，每个硬币要么是正面（用 '1' 表示）要么是反面（用 '0' 表示）。

每次操作，小 L 可以选择两个 **相邻** 的硬币，同时翻转它们（即正面变反面，反面变正面）。

小 L 想知道，是否可以通过若干次这样的操作，使得 **所有硬币都变成正面** 。

输入格式

一行一个只包含字符 '0' 和 '1' 的字符串 s（$2 \leq |s| \leq 20$），表示排成一行的每个硬币的状态。

输出格式

输出一行一个字符串，如果可以使所有硬币变为正面，输出 "Yes" ，否则输出 "No" 。

你可以以任意大小写输出 "Yes" 和 "No"（例如，字符串 "yEs"、"yes"、"Yes" 和 "YES" 将被识别为肯定的回答）。

样例

standard input	standard output
00	Yes
010	Yes
101	No
1111	Yes

提示说明

对于第一个样例，只需要选择第 1 个硬币和第 2 个硬币翻转，然后所有硬币都变成正面。

对于第二个样例，第一次选择第 1 个硬币和第 2 个硬币翻转，第二次选择第 2 个硬币和第 3 个硬币翻转，然后所有硬币都变成正面。

对于第三个样例，可以证明无论如何操作，都无法使得所有硬币都变成正面。

对于第四个样例，所有硬币都已经都是正面。

Problem M. 字符消消乐

输入文件: standard input
输出文件: standard output

小 L 最近又迷上了一款字符消消乐游戏，其规则如下：

在字符串中，如果有 **相邻且相同** 的一段 **长度至少为** 3 的子字符串，就可以同时消除这些字符，消除后剩下的部分会 **前后拼接** 在一起（例如，“abbbcc” 消除 “bbb” 后得到 “acc”）。

小 L 至多只能进行 k 次消除操作，他想知道，他 **最短** 能将字符串缩短到多短？

小 L 觉得这个问题太难了，于是从雪碧那里获得了一个作弊魔法，他可以在游戏开始前 **任意排列** 字符串中的字符顺序。

输入格式

第一行输入两个整数 n 和 k（$1 \le n \le 10^5$，$0 \le k \le 10^5$），分别表示字符串的长度和最多消除次数。

第二行输入一个长度为 n 的字符串 s，字符串由小写字母组成。

输出格式

输出一个整数，表示进行至多 k 次消除后字符串可能的最小长度。

样例

standard input	standard output
9 3 aaccbbbba	2
5 1 aaaaa	0
8 4 abacadae	4

提示说明

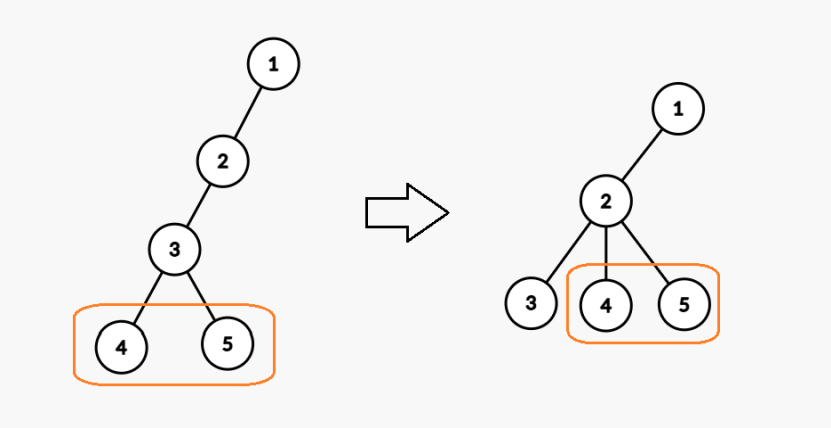
在第一个样例中，可以重新排列为 “aaabbbbcc” ，第一次消除 “bbbb” 得到 “aaacc” ，第二次消除 “aaa” 得到 “cc”，最终长度为 2 。

Problem N. 缩花

输入文件: standard input
输出文件: standard output

给定一棵包含 n 个节点的树，你需要对其执行若干次操作，每次操作按如下顺序进行：

- 指定一个点 r 作为根节点，此次操作后续步骤均以 r 为根节点。
- 选择一个非根节点 u ，将 u 的所有直接儿子节点连向 u 的边断开，然后连接到 u 的父亲节点上。



定义菊花图为不存在任何长度大于 3 的路径的图，你的目标是通过最少的操作将给定的树变为菊花图，找到可能的 **最少** 操作次数。

如果至少需要一次操作，你还需要输出最优操作序列（满足操作次数最少）的第一次操作选择的 r 和 u 。如果存在多个最优操作序列，输出任意一组 r 和 u 即可。

输入格式

第一行输入一个整数 t（$1 \le t \le 10^4$），表示测试用例数量。

对于每个测试用例：

- 第一行一个整数 n $(1 \le n \le 10^5)$，表示树的节点数。
- 接下来 $n-1$ 行，每行两个整数 u, v $(1 \le u, v \le n)$，表示树的一条边。

数据保证所有测试用例的 n 总和不超过 10^5。

输出格式

对于每个测试用例：

- 第一行输出一个整数，表示最小的操作次数。
- 如果至少需要一次操作，在下一行输出两个整数 r 和 u（$1 \le r, u \le n$ 且 $r \ne u$），表示第一次操作选择的根节点 r 和被操作的节点 u。

样例

standard input	standard output
3 2 1 2 4 1 2 1 3 1 4 5 1 2 2 3 3 4 3 5	0 0 1 1 3
1 8 1 2 1 3 2 4 2 5 3 6 3 7 3 8	2 1 3