第四届“梦羽杯”程序设计大赛

正式赛

2026 年 4 月 25 日

试题列表

A	欢迎来到梦羽杯
B	圆域判定
C	图上染色
D	三合一
E	曼哈顿之心
F	人类的本质是复读机
G	交换区间和
H	专项 Rating 系统
I	石子游戏
J	金字塔
K	邻数逻辑
L	最小差值

本试题册共 12 题，18 页。

如果您的试题册缺少页面，请立即通知志愿者。

Problem A. 欢迎来到梦羽杯

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

屏幕前的同学，你好！欢迎来到第四届梦羽杯程序设计大赛的现场。
在传统的期末考试中，你很难立刻知道自己做对没有，这往往让人感到焦虑。但在算法竞赛的世界里，评测机是绝对诚实且即时的。当你提交代码后，系统会立刻给出反馈。屏幕上亮起的那一抹绿色的“Accepted”，就是计算机对你逻辑的最高赞美。
算法竞赛是一项严谨的智力运动，你的每一次提交都会被记录在案，并在排行榜上留下痕迹。因此，深思熟虑的逻辑和严谨的代码是我们共同追求的目标。但这并不意味着你必须做到完美无缺才敢动手。
请记住，算法竞赛不是一考定终身的答卷，而是一个允许你不断试错、不断接近完美的实验室。所以，不要害怕提交。现在，用这道最简单的题，大胆地向评测机发起你的第一次提交吧！
作为整场比赛的开端，本题不需要提供任何输入。请你编写一个程序，直接输出你对这场比赛最美好的期许：We will get Accepted!

Input

本题没有输入数据。

Output

输出一行字符串：We will get Accepted!

Examples

standard input	standard output
	We will get Accepted!

Problem B. 圆域判定

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定平面直角坐标系中的圆心坐标 (x_c, y_c)、圆半径 r 以及一个点的坐标 (x_p, y_p)。判定该点与圆的位置关系：

- 若点在圆内，输出 `Inside`；
- 若点在圆上，输出 `On`；
- 若点在圆外，输出 `Outside`。

Input

第一行输入一个正整数 T $(1 \le T \le 10^5)$，表示测试用例的数量。
接下来的 T 行，每行包含五个整数 x_c, y_c $(-10^9 \le x_c, y_c \le 10^9)$，$r$ $(1 \le r \le 10^9)$，x_p, y_p $(-10^9 \le x_p, y_p \le 10^9)$，分别代表圆心横纵坐标、圆半径以及点的横纵坐标。

Examples

standard input	standard output
4 0 0 5 3 4 0 0 5 0 0 0 0 5 6 6 1 1 2 1 3	On Inside Outside On

Problem C. 图上染色

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定一个包含 N 个顶点与 M 条边的有向图，无自环且无重边。所有顶点的初始颜色均为白色。请按顺序处理 Q 次操作，操作包含以下两种类型：

- `1 v`：将顶点 v 染成黑色。
- `2 v`：判断是否存在一条从顶点 v 出发，且能到达任意一个黑色顶点的路径。

Input

第一行输入两个整数 N $(1 \le N \le 2 \times 10^5)$ 和 M $(0 \le M \le 2 \times 10^5)$，分别表示图的顶点数与有向边数。

接下来的 M 行，每行输入两个正整数 X_i $(1 \le X_i \le N)$ 和 Y_i $(1 \le Y_i \le N)$，表示存在一条从 X_i 指向 Y_i 的有向边。

接下来的一行输入一个正整数 Q $(1 \le Q \le 2 \times 10^5)$，表示操作的总次数。

接下来的 Q 行，每行输入一次操作，格式为 `1 v` 或 `2 v` $(1 \le v \le N)$。

Output

对于每次 `2 v` 操作，输出一行字符串。若存在到达黑色顶点的路径输出 `Yes`，否则输出 `No`。

Examples

standard input	standard output
5 6 1 2 2 3 3 1 4 5 1 4 2 5 5 1 3 2 1 2 4 1 5 2 4	Yes No Yes

Note

对于样例 1：

- 第一次操作：将顶点 3 染黑。

- 第二次操作：存在路径 $1 \to 2 \to 3$，顶点 1 可到达黑色顶点 3，输出 `Yes`。
- 第三次操作：从顶点 4 无法到达唯一的黑色顶点 3，输出 `No`。
- 第四次操作：将顶点 5 染黑。此时黑色顶点为 3 和 5。
- 第五次操作：存在边 $4 \to 5$，顶点 4 可到达黑色顶点 5，输出 `Yes`。

Problem D. 三合一

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定三个整数 A、B 和 C，计算它们的和。

Input

输入包含一行三个整数 A $(-10^8 \le A \le 10^8)$，B $(-10^8 \le B \le 10^8)$，C $(-10^8 \le C \le 10^8)$，相邻整数之间以单个空格分隔。

Output

输出一个整数，表示 $A + B + C$ 的计算结果。

Examples

standard input	standard output
100 200 300	600

Problem E. 曼哈顿之心

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定二维平面上的 n 个点，坐标可能重合。
请找出存在多少个整数坐标点 (X, Y)，使得给定的 n 个点到 (X, Y) 的曼哈顿距离之和最小。
任意两点 (x_1, y_1) 和 (x_2, y_2) 之间的曼哈顿距离定义为 $|x_1 - x_2| + |y_1 - y_2|$。

Input

第一行输入一个正整数 t $(1 \le t \le 500)$，表示测试用例的数量。
对于每组测试用例，第一行输入一个正整数 n $(1 \le n \le 1000)$，表示点的数量。
接下来的 n 行，每行输入两个整数 x_i, y_i $(0 \le x_i, y_i \le 10^9)$，表示每个点的横纵坐标。

Output

对于每组测试用例，输出一行一个整数，表示满足曼哈顿距离之和最小的整数坐标点的个数。

Examples

standard input	standard output
6 3 0 0 2 0 1 2 4 1 0 0 2 2 3 3 1 4 0 0 0 1 1 0 1 1 2 0 0 1 1 2 0 0 2 0 2	1 4 4 4 3 1

standard input	standard output
0 0 0 0	

Problem F. 人类的本质是复读机

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定一个压缩字符串，该字符串由若干连续的单元拼接而成，每个单元均由一个小写英文字母及其后紧跟的一个正整数组成。 请将该字符串还原，把每个小写字母按其后紧跟的数字连续复制相应次数，输出完整展开后的原始字符串。

Input

输入一行，包含压缩字符串 S $(1 \le |S| \le 1000)$。

- 字符串 S 仅由小写英文字母与数字组成。
- 每一个数字代表其紧邻的左侧字母的重复次数 k $(1 \le k \le 100)$。

Output

输出一行，表示展开后的原始字符串。

Examples

standard input	standard output
a3b10c1	aaabbbbbbbbbbc
x2y2z2	xxyyzz

Problem G. 交换区间和

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定一个长度为 N 的序列 $A = (A_1, A_2, ..., A_N)$。按顺序处理 Q 次操作，操作分为以下两种：

- `1 x`：交换 A_x 与 A_{x+1} 的值。
- `2 l r`：求 $\sum_{i=l}^{r} A_i$ 的值。

Input

第一行输入两个正整数 N $(2 \le N \le 10^5)$ 和 Q $(1 \le Q \le 10^5)$，分别表示序列长度与操作次数。
第二行输入 N 个正整数 $A_1, A_2, ..., A_N$ $(1 \le A_i \le 10^4)$，表示初始序列。
接下来的 Q 行，每行输入一次操作，格式为 `1 x` $(1 \le x \le N-1)$ 或 `2 l r` $(1 \le l \le r \le N)$。

Output

对于每次 `2 l r` 操作，输出一行一个整数，表示该区间和的值。

Examples

standard input	standard output
4 4 2 7 1 8 1 2 2 1 2 1 1 2 2 4	3 17
8 10 22 75 26 45 72 81 47 29 2 2 7 2 6 8 2 4 4 1 2 2 1 3 1 1 2 2 4 1 2 1 4 2 1 1	346 157 45 123 142 26

Problem H. 专项 Rating 系统

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

对于刚接触算法竞赛（Competitive Programming）的萌新来说，Codeforces 和 AtCoder 这样的平台就像是充满挑战的修罗场。在这里，选手们的综合实力被量化为一个极具分量的核心指标——Rating（**等级分**）。一场比赛分数的涨跌，不仅决定了你 ID 的颜色，更是对你近期训练成果的直接反馈。

然而，随着刷题的深入，许多选手都会碰壁，并发现这套成熟的 Rating 系统其实有一个致命的盲区：它是高度“一刀切”的。在真实的竞赛生态中，几乎没有人是全能的。一个 Rating 1500 的选手，可能在“动态规划（dp）”上思维极其敏锐，能越级手撕 1900 分的难题；但在“图论（graphs）”面前，却常常被击穿防线。这种普遍的“偏科”现象，全局 Rating 根本无法精准刻画。

既然全局 Rating 看不出单项的短板，那最自然的解决思路就是：**拆开算，给每一个算法标签（Tag）建立独立的分数**。但这马上就会引出一个不可回避的逻辑难题——现实中的竞赛题目，极少只考察单一纯粹的知识点。一道有区分度的题目，往往是由多个知识模块交织而成的“缝合怪”，带有 `dp`、`math`、`trees` 等多个标签。

基于此，你需要为自己开发一套“专项 Rating 系统”，用来跟踪你在各个算法标签上的 Rating。

【系统初始化】

在系统启动时，你会预先输入 N 个算法标签及你在这些标签上的初始 Rating。需要注意的是，本系统仅处理这 N 个预设标签，后续操作中涉及的所有标签均保证属于这 N 个已知标签。

【操作指令】

初始化完成后，系统会持续接收并处理以下五类操作指令：

- `1 K D T O Tag_1 Tag_2 ... Tag_K`：**记录一次刷题结算**。完成难度为 D、耗时 T、结果为 O（$O = 1$ 为通过，$O = 0$ 为未通过）的题目，涉及 K 个标签。
- `2 x`：**环境调整**。由于题目难度的“通货膨胀”，老分数的含金量会不断缩水。将系统中当前所有标签的真实 Rating 统一**减去**正整数 x。
- `3`：**查询当前最强项**。查询目前系统中分数最高的标签（若有多个最高分，取字典序最小的那个）。
- `4`：**查询致命短板**。查询目前系统中分数最低的标签（若有多个最低分，取字典序最小的那个）。
- `5 Tag`：**单项诊断**。查询系统当前关于某特定 `Tag` 的真实 Rating。

核心结算规则

针对指令 `1` 中的每一个标签（当前评分为 R），按以下两步顺序更新：

第一步：计算分差系数 X

1. 计算初始分差：$X_{raw} = \frac{D-R}{25}$ （结果向零取整）。
2. 确定最终系数：$X = \max(-16, \min(16, X_{raw}))$。

第二步：计算变动量 Δ_{score}

- **情况 1：通过题目** $(O = 1)$

$$\Delta_{score} = 16 + X$$

逻辑： 成功 AC 时，系统根据题目难度与个人水平的差值给予分数奖励。越级挑战成功将获得高额加分，而做出低难度题的收益将被极大程度地削减。

- **情况 2：未通过但有深度尝试** $(O = 0$ **且** $T \geq 30)$

$$\Delta_{score} = -16 + X$$

逻辑： 未能通过题目且耗时较长。此时系统会执行一次惩罚性的扣分，扣除额度同样受难度差值 X 修正：挑战难题时的扣分较少，而未能通过简单题则会面临更重的分数处罚。

- **情况 3：未通过且放弃较快** $(O = 0$ **且** $T < 30)$

$$\Delta_{score} = 0$$

逻辑： 30 分钟内放弃且未通过，系统判定此次尝试属于无效练习，不执行任何分数结算。

结算完成后，该项标签 Rating **更新为：** $R \leftarrow R + \Delta_{score}$。

Input

第一行输入两个正整数 N, Q $(1 \leq N, Q \leq 10^3)$，分别代表系统初始化的标签总数和随后的操作总数。

接下来 N 行，每行包含一个字符串 Tag_i 和一个整数 R_i $(0 \leq R_i \leq 4000)$，代表标签名称及其初始 Rating。

- 标签仅由小写英文字母组成，长度不超过 20。
- 保证 N 个初始标签名称各不相同。

接下来 Q 行，每行代表一个操作，格式如下：

`1 K D T O Tag_1 Tag_2 ... Tag_K`：记录一次刷题结算。K 为涉及的标签数 $(1 \leq K \leq \min(N, 5))$，$D$ 为题目难度，T 为耗时（分钟），O 为结果 (1 为通过，0 为未通过)，随后是 K 个标签名。

`2 x`：环境调整。读入一个正整数 x $(1 \leq x \leq 100)$，将当前系统中所有标签的真实 Rating 统一减去 x。

`3`：查询当前最强项。输出当前真实 Rating 最高且字典序最小的标签名称及其分数，空格分隔。

`4`：查询致命短板。输出当前真实 Rating 最低且字典序最小的标签名称及其分数，空格分隔。

`5 Tag`：单项诊断。输出指定标签当前的真实 Rating 分数。

Output

针对指令 3、4、5，每行输出一个对应的结果。

Examples

standard input	standard output
2 6 dp 1500 math 1500 1 2 1600 40 1 dp math 3 2 10 5 dp 1 1 1200 10 0 dp 4	dp 1520 1510 dp 1510

Note

针对 $X_{raw} = \frac{D-R}{25}$ 向零取整的代码实现参考：

```
// C++ / Java
int x_raw = (D - R) / 25;

// Python
x_raw = int((D - R) / 25)
```

Problem I. 石子游戏

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定 n 堆石子，第 i 堆包含 a_i 个石子。两名玩家轮流进行游戏，规则如下：

- 每次操作只能从编号最小的非空石堆中取走任意正整数个石子。
- 无法进行操作者（即所有石堆均为空时）判负。

假设双方均采取最优策略，判定先手获胜还是后手获胜。

Input

第一行输入一个正整数 t $(1 \le t \le 1000)$，表示测试用例的数量。

接下来的每组测试用例中，第一行输入一个正整数 n $(1 \le n \le 10^5)$，表示石堆的数量，且保证所有测试用例中 n 的总和不超过 10^5。

每组测试用例的第二行输入 n 个正整数 $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^9)$，依次表示各堆石子的数量。

Output

对于每组测试用例，输出一行字符串。若先手获胜输出 `First`，若后手获胜输出 `Second`。

Examples

standard input	standard output
7 3 2 5 4 8 1 1 1 1 1 1 1 1 6 1 2 3 4 5 6 6 1 1 2 1 2 2 1 1000000000 5 1 2 2 1 1 3 1 1 1	First Second Second First First Second First

Note

在第一个测试用例中，初始状态为 $[2, 5, 4]$，先手的必胜策略如下：

- 先手从第一堆取走 1 个石子，状态变为 $[1, 5, 4]$。

- 后手只能从第一堆取走仅剩的 1 个石子，状态变为 $[0, 5, 4]$。
- 先手从第二堆取走 4 个石子，状态变为 $[0, 1, 4]$。
- 后手只能从第二堆取走仅剩的 1 个石子，状态变为 $[0, 0, 4]$。
- 先手从第三堆取走 4 个石子，状态变为 $[0, 0, 0]$。
- 后手面临全部为空的石堆，无法操作，先手获胜。

Problem J. 金字塔

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

定义尺寸为 k 的金字塔数列为长度为 $2k-1$ 的数列，其元素依次为 $1, 2, ..., k-1, k, k-1, ..., 2, 1$。

给定一个长度为 N 的数列 $A = (A_1, A_2, ..., A_N)$。你可以对该数列执行以下操作任意次（可以为 0 次）：

- 选择数列中的任意一项，使其值减 1。
- 删除数列的首项或末项。

求通过操作能够得到的金字塔数列的最大尺寸 k。

Input

第一行输入一个正整数 N $(1 \le N \le 2 \times 10^5)$，表示数列的长度。

第二行输入 N 个正整数 $A_1, A_2, ..., A_N$ $(1 \le A_i \le 10^9)$，表示初始数列的元素。

Output

输出一个整数，表示能够得到的金字塔数列的最大尺寸。

Examples

standard input	standard output
5 2 2 3 1 1	2
5 1 2 3 4 5	3
1 1000000000	1

Note

对于样例 1，可通过以下操作得到尺寸为 2 的金字塔数列 $(1, 2, 1)$：

- 将第 3 项减 1，数列变为 $(2, 2, 2, 1, 1)$。
- 删除首项，数列变为 $(2, 2, 1, 1)$。
- 删除末项，数列变为 $(2, 2, 1)$。
- 将第 1 项减 1，数列变为 $(1, 2, 1)$。

Problem K. 邻数逻辑

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定一个整数 x。请分别判定是否存在非负整数 a，使得以下三个等式成立：
- $x = a \,\&\, (a+1)$
- $x = a \mid (a+1)$
- $x = a \oplus (a+1)$

其中 $\&, \mid, \oplus$ 分别代表按位与、按位或、按位异或运算。

Input

第一行输入一个正整数 T $(1 \le T \le 10^5)$，表示测试用例的数量。
接下来的 T 行，每行包含一个整数 x $(0 \le x \le 10^{18})$，表示给定的目标数值。

Output

对于每组测试用例，输出一行三个字符串，依次对应 $\&, \mid, \oplus$ 运算的判定结果。若等式成立输出 `Yes`，否则输出 `No`。三个字符串之间以单个空格分隔。

Examples

standard input	standard output
2 0 1	Yes No No No Yes Yes

Note

- 当 $x = 0$ 时：存在 $a = 0$ 使得 $0 = 0 \,\&\, 1$，但不存在非负整数 a 满足或、异或等式结果为 0。
- 当 $x = 1$ 时：存在 $a = 0$ 使得 $1 = 0 \mid 1$ 且 $1 = 0 \oplus 1$，但不存在非负整数 a 满足与等式结果为 1。

位运算是将数字转换为二进制后，对每一位（0 或 1）独立进行的运算。以下为运算规则及示例（以二进制位数对齐对比）：

- **按位与** (&)：两位同时为 1 时，结果才为 1，否则为 0。
  - 示例：5 & 3
  - 5 的二进制：`101`
  - 3 的二进制：`011`
  - 结果运算：`001`（转化为十进制为 1）
- **按位或** (|)：两位中只要有一个为 1，结果就为 1；同为 0 才为 0。

- 示例：$5 \mid 3$
- 5 的二进制：101
- 3 的二进制：011
- 结果运算：111（转化为十进制为 7）

- **按位异或** ($\oplus$)：两位**不相同**时结果为 1，相同则为 0。
  - 示例：$5 \oplus 3$
  - 5 的二进制：101
  - 3 的二进制：011
  - 结果运算：110（转化为十进制为 6）

Problem L. 最小差值

Input file:	standard input
Output file:	standard output
Time Limit:	2 second
Memory Limit:	512 megabytes

给定长度分别为 N 和 M 的正整数数列 A 和 B。求 $\min_{1\le i\le N,1\le j\le M}|A_i-B_j|$。

Input

第一行输入两个正整数 N $(1\le N\le 2\times 10^5)$ 和 M $(1\le M\le 2\times 10^5)$，分别表示数列 A 和 B 的长度。
第二行输入 N 个正整数 $A_1,A_2,...,A_N$ $(1\le A_i\le 10^{18})$，表示数列 A 的元素。
第三行输入 M 个正整数 $B_1,B_2,...,B_M$ $(1\le B_i\le 10^{18})$，表示数列 B 的元素。

Output

输出一个整数，表示两个元素之差的绝对值的最小值。

Examples

standard input	standard output
2 2 1 6 4 9	2
1 1 10 10	0
6 8 82 76 82 82 71 70 17 39 67 2 45 35 22 24	3