2025 icpc 国际大学生程序设计竞赛
亚洲区域赛（南京站）

正式赛

2025 年 11 月 9 日

试题列表

A	哇，昨日六次重现
B	什么，还有袋鼠？
C	分糖果
D	秋子和高尔夫
E	青白树
F	按位与路径
G	水桶盛宴
H	Pen Pineapple Apple Pen
I	吃饭
J	Trajan 算法
K	象棋
L	区域赛冠军
M	很多凸多边形

本试题册共 13 题，20 页。
如果您的试题册缺少页面，请立即通知志愿者。

承办方

命题方

竞赛过程中访问非竞赛网页是违反竞赛规则的行为。
如果您有兴趣（我们很荣幸），
请在竞赛后扫描二维码。

Problem A. 哇，昨日六次重现

这是一道交互题。

继 2018 至 2024 年成功承办赛事之后，南京航空航天大学（NUAA）将连续第八年承办国际大学生程序设计竞赛（ICPC）。

在 2018 与 2019 年，“中二之力” 队与 “三个顶俩” 队为清华大学赢得了冠军。在 2020，2021 与 2022 年，北京大学的 “逆十字” 队赢得三连冠。在 2023 年，来自北京大学的另一支队伍 “重生之我是菜狗” 赢得了冠军。在 2024 年，浙江大学的 “来生” 队摘得桂冠。

今年，将会有约 335 支队伍参与南京站的竞赛。本次竞赛将会颁发至多 33 项金奖，66 项银奖与 99 项铜奖（数字仅供参考）。让我们期待选手们出色的表现！我们还想要感谢竞赛组委会与志愿者们的努力付出。感谢你们为本次竞赛做出的贡献！

在 2023 ICPC 国际大学生程序设计竞赛亚洲区域赛（南京站）中拍摄的照片

在 2018 年的竞赛中，K 题《袋鼠谜题》要求选手为以下游戏构造一个操作序列：

> 谜题由一个 n 行 m 列的网格（$1 \le n, m \le 20$）组成，且有一些（至少 2 只）袋鼠位于网格中。玩家的目标是控制袋鼠并把它们聚集在同一个格子中。一些格子里有墙，袋鼠无法进入这些有墙的格子，而其它格子是空的。袋鼠可以从一个空格子移动到上，下，左，右相邻的另一个空格子中。
>
> 游戏开始时，每个空格子里都有一只袋鼠。玩家可以通过键盘上 U，D，L，R 四个按键控制袋鼠的移动。所有袋鼠会同时根据您按下的按键移动。
>
> 选手需要构造一个长度至多为 5×10^4 且由 U，D，L，R 组成的操作序列以达成目标。

在 2020 年的竞赛中，A 题《啊，昨日重现》要求选手构造一张输入地图，以证明以下代码并不是上述问题的解：

```cpp
#include <bits/stdc++.h>
using namespace std;
string s = "UDLR";
int main()
{
  srand(time(NULL));
  for (int i = 1; i <= 50000; i++) putchar(s[rand() % 4]);
  return 0;
}
```

此外，在 2021 年的竞赛（A 题，《呀，昨日再次重现》），2022 年的竞赛（A 题，《停停，昨日请不要再重现》）2023 年的竞赛（A 题，《酷，昨日四次重现》），和 2024 年的竞赛（A 题，《嘿，有看到我的袋鼠吗?》）中，每年都有一道与袋鼠相关的问题！我们很想向您介绍所有这些问题，但如果我们每年都这样做，在 3025 年的竞赛中将会有一道题拥有长达 500 页的题面。因此，这次我们省略它们。另外，您可能已经在热身赛中见过它们了。

在 2025 年的竞赛中，如大家期待的那样，袋鼠题又回来啦！我们不知道为什么命题组的成员们那么喜欢袋鼠，但题目如下：

给定一张 n 行 m 列的网格，在位于第 i_h 行第 j_h 列的格子上有一个洞，其它每个格子都是空地并且都有一只袋鼠。

相似地，袋鼠可以被键盘上的 U，D，L，R 键控制。所有袋鼠会同时根据按下的按键移动。具体来说，对于一只位于第 i 行第 j 列的格子（用 (i, j) 表示）上的袋鼠：

1. 按键 U：它会移动到 $(i-1, j)$。
2. 按键 D：它会移动到 $(i+1, j)$。
3. 按键 L：它会移动到 $(i, j-1)$。
4. 按键 R：它会移动到 $(i, j+1)$。

如果一只袋鼠踩到了洞（也就是说，$i = i_h$ 且 $j = j_h$）或者移动到了网格外面，它将被从网格上移除。

问题在于，i_h 与 j_h 的值是未知的。您需要在至多 $(n + m + 10)$ 次询问内确定洞的位置。每次询问，您可以按一个按键（U，D，L，或 R）。裁判程序会输出一个整数 t 作为询问的答案，表示在按键后，网格上还剩几只袋鼠。

请注意：裁判程序不是适应性的。也就是说，每组测试数据的答案是事先确定的。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 100$）表示测试数据组数，对于每组测试数据：

第一行输入两个整数 n 和 m（$3 \le n, m \le 30$），表示网格的行数和列数。

Interaction Protocol

要提出询问，请输出一行。首先输出 `?`，之后跟一个空格，然后输出一个英文字母（U，D，L，或 R）。在清空输出缓冲区之后，您的程序需要读入一个整数 t，表示对您的询问的回答。请回忆：对于每组数据，您至多可以提出 $(n + m + 10)$ 次询问。

要猜测洞的位置，请输出一行。首先输出 `!`，之后跟一个空格，然后输出两个由单个空格分隔的整数 i_h 和 j_h（$1 \le i_h \le n, 1 \le j_h \le m$）表示洞的位置。在清空输出缓冲区之后，您的程序应该马上开始处理下一组测试数据。如果没有更多测试数据，您的程序应该立即退出。还请注意，猜测不算一次询问。

清空输出缓冲区可以使用以下方式：

- C 和 C++ 使用 `fflush(stdout)`（如果您使用 `printf`）或 `cout.flush()`（如果您使用 `cout`）。
- Java 和 Kotlin 使用 `System.out.flush()`。
- Python 使用 `stdout.flush()`。

Example

standard input	standard output
2 3 4 7 3 4 3 8 5	 ? L ? D ! 2 3 ? U ? R ! 1 1

Problem B. 什么，还有袋鼠？

一条数轴上有 n 只袋鼠，其中第 i 只袋鼠最初位于正坐标 c_i（即 $c_i > 0$），并且有两种步长 a_i 和 b_i。

袋鼠可以被键盘上的 1, 2, 3, 4 键控制。所有袋鼠会同时根据按下的按键移动。具体来说，令 x_i 表示第 i 只袋鼠的当前坐标：

- 按键 1：它会移动到 $(x_i + a_i)$。
- 按键 2：它会移动到 $(x_i - a_i)$。
- 按键 3：它会移动到 $(x_i + b_i)$。
- 按键 4：它会移动到 $(x_i - b_i)$。

允许两只或更多袋鼠位于相同坐标。若两只袋鼠在移动过程中相遇，它们将互相穿过。

您可以按键任意多次。最大化操作后具有负坐标的袋鼠的数量（即最大化满足 $x_i < 0$ 的袋鼠的数量）。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 10^5$）表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n（$1 \le n \le 2 \times 10^5$）表示袋鼠的数量。

对于接下来 n 行，第 i 行输入三个整数 a_i，b_i 和 c_i（$-10^9 \le a_i, b_i \le 10^9$，$1 \le c_i \le 10^9$），描述了第 i 只袋鼠。

保证所有数据 n 之和不超过 2×10^5。

Output

每组数据输出一行一个整数，表示操作后具有负坐标的袋鼠的最大数量。

Example

standard input	standard output
2 4 1 1 1 -1 3 4 0 -2 7 -5 0 10 1 0 0 1	3 0

Problem C. 分糖果

堡堡和梦格是经常一起分享糖果的好朋友。堡堡现在有 a 颗糖果，梦格有 b 颗糖果，即一共有 $n = a + b$ 颗糖果。

不仅如此，堡堡还发现，这 n 颗糖果可以被分成 a 堆，每堆都有相同数量的糖果。同时梦格也发现，这 n 颗糖果也可以被分成 b 堆，每堆也有相同数量的糖果。

将糖果混合后，他们忘记了原来每个人有几颗糖果。请帮助他们找到一对可能的正整数 a 和 b，或表明这是不可能的。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 10^5$）表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n（$1 \le n \le 10^{18}$）表示糖果的总数。

Output

对于每组数据：

- 若存在这样的正整数 a 和 b，首先输出一行 `Yes`，之后输出一行两个由单个空格分隔的整数 a 和 b。若有多种合法答案，您可以输出任意一种。
- 否则，若不存在这样的正整数 a 和 b，只要输出一行 `No`。

Example

standard input	standard output
3 1 6 7	No Yes 3 3 No

Problem D. 秋子和高尔夫

秋子酷爱高尔夫运动，但奈何球技一般。

秋子为了练习球技，在家里买了 k 个室内高尔夫推杆练习器。和普通的练习器不同，他买的练习器很大。具体来说，每个练习器可以抽象成一棵有 n_i 个节点的树，其中节点 1 是球洞。每个练习器上都有一颗高尔夫球。对于第 i 个练习器，球一开始位于节点 s_i。

每次击球，秋子会选择一个练习器。由于他的水平有限，这颗高尔夫球会被等概率击打到相邻节点上。若球进入洞（节点 1）中，则结束今天的练习。否则，秋子将重复以上过程，直到将某颗球击入洞中。

秋子的目标是最小化结束练习所需的期望击球次数。若每次击球他都做出最优选择，求期望击球次数。

Input

每个测试文件仅有一组测试数据。

第一行输入一个整数 k（$1 \le k \le 300$），表示高尔夫推杆练习器的个数。

接下来输入对各个练习器的描述。对于每个练习器：

- 第一行输入两个整数 n_i 和 s_i（$2 \le s_i \le n_i \le 300$）。
- 对于接下来 $(n_i - 1)$ 行，第 j 行输入两个整数 $u_{i,j}$ 和 $v_{i,j}$（$1 \le u_{i,j}, v_{i,j} \le n_i$），表示有一条边连接节点 $u_{i,j}$ 和 $v_{i,j}$。保证这些边形成了一棵树。

Output

输出一行一个数，表示在最优策略下的期望击球次数。

如果您的答案的绝对误差或相对误差不超过 10^{-6}，则将被视为正确。更正式地，假设您的输出为 a，标准答案为 b，当且仅当 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$ 时，您的输出才会被接受。

Example

standard input	standard output
2 4 2 1 2 2 3 3 4 4 3 1 2 2 3 3 4	4.666666666666666

Problem E. 青白树

给定一棵有 n 个节点的树，每个节点的颜色为青色或白色。对于树中的一条简单路径，设 c 为路径中青色节点的数量，w 为路径中白色节点的数量。定义路径的价值为 $(c+w)-2\times|c-w|$，其中 $|c-w|$ 是 $(c-w)$ 的绝对值。

定义节点 u 的深度为从节点 1 到节点 u 的最短路径上的边数。定义一条简单路径上的关键点为路径上所有节点中深度最小的节点。可以证明一条简单路径的关键点是唯一的。

对于每个 $1\le i\le n$，求所有关键点为 i 的路径的最大价值。

请回忆：树中的路径是一个非空的节点序列 $p_1,p_2,\ldots,p_k$，序列中每对相邻节点之间都有一条边连接。简单路径是一条所有节点互不相同的路径。单个节点也可以形成简单路径。

Input

有多组测试数据。第一行输入一个整数 T（$1\le T\le 10^5$）表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n（$2\le n\le 4\times 10^5$），表示节点的数量。

第二行输入一个长度为 n 的 01 字符串，表示每个节点的颜色。如果第 i 个字符是 '0'，则第 i 个节点为青色；否则，第 i 个节点为白色。

对于接下来 $(n-1)$ 行，第 i 行输入两个整数 u_i 和 v_i（$1\le u_i,v_i\le n$），表示一条连接节点 u_i 和 v_i 的边。保证给定的边形成一棵树。

保证所有数据 n 之和不超过 4×10^5。

Output

每组数据输出 n 行，每行包含一个整数。第 i 行表示所有关键点为 i 的路径的最大价值。

Example

standard input	standard output
3 8 00101110 1 2 1 3 3 4 3 5 5 6 5 7 7 8 2 01 1 2 2 00 1 2	6 -1 3 -1 1 -1 2 -1 2 -1 -1 -1

Problem F. 按位与路径

给定一张 n 个节点的无向图，一开始图中没有边。维护 q 次操作，操作可能是以下两种：

- $+\ u\ v\ w$：在节点 u 和 v 之间连一条边，边权为 w。
- $?\ u\ v$：定义路径的权值为，路径上所有边权进行按位与运算的结果。求以节点 u 为起点，节点 v 为终点的路径的最大权值。若不存在连接节点 u 和 v 的路径，则答案为 -1。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 100$）表示测试数据组数，对于每组测试数据：

第一行输入两个整数 n 和 q（$2 \le n \le 10^3$，$1 \le q \le 10^6$）表示节点的数量和操作的数量。

对于接下来 q 行，第 i 行首先输入一个字符 op（$op \in \{+, ?\}$）表示第 i 次操作的类型。

- 若 $op = +$，则继续输入三个整数 u，v 和 w（$1 \le u, v \le n$，$0 \le w < 2^{12}$），表示第一种操作。
- 若 $op =?$，则继续输入两个整数 u 和 v （$1 \le u, v \le n$，$u \ne v$），表示第二种操作。

保证每组数据至少有一次第二种操作。另外，所有数据的 n 之和不超过 10^3，q 之和不超过 10^6。

Output

为了减少输出的大小，每组数据只需要输出一行一个整数，表示所有第二种操作的答案之和。

Example

standard input	standard output
2 4 11 + 1 3 6 + 3 4 2 ? 1 4 + 1 4 3 ? 4 1 + 4 3 4 ? 1 4 + 3 2 3 ? 1 2 + 1 1 4 ? 1 3 4 4 + 1 2 64 + 3 4 32 ? 1 2 ? 3 1	17 63

Note

对于第一组样例数据，第二种操作的答案分别为 2，3，4，2 和 6。

对于第二组样例数据，第二种操作的答案分别为 64 和 -1。

Problem G. 水桶盛宴

欢迎来到 Mr. Ant Tenna 的电视节目时间！在本轮游戏中，您将获得 n 个水桶，其中第 i 个水桶的最大容量为 v_i。游戏开始时，所有水桶都会被注满水，您的任务就是在接下来的 t 秒内尽可能地保持水量。但这里有个问题：这些水桶是坏的！第 i 个水桶的漏水速度是 l_i，也就是说，游戏开始后的每秒钟，第 i 个水桶会漏掉 l_i 体积的水。

在游戏开始前，您可以将任意数量的水桶合并成一个。合并后的水桶容量等于被合并水桶里的最大容量，漏水速度等于被合并水桶的最小速度。您可以进行任意多次合并操作，但只能在注水之前合并水桶。

Tenna 提出了 q 个问题，其中第 i 个问题的时间限制是 t_i。对于每个 $1 \le i \le q$，求 t_i 秒后最多共能保留多少体积的水。请注意：每个问题是独立的。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 5 \times 10^4$）表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n（$1 \le n \le 2 \times 10^5$），表示水桶的数量。

第二行输入 n 个整数 $v_1, v_2, \cdots, v_n$（$1 \le v_i \le 4 \times 10^{13}$），其中 v_i 表示第 i 个水桶的容量。

第三行输入 n 个整数 $l_1, l_2, \cdots, l_n$（$0 \le l_i \le 10^9$），其中 l_i 表示第 i 个水桶的漏水速度。

第四行输入一个整数 q（$1 \le q \le 2 \times 10^5$），表示问题的数量。

第五行输入 q 个整数 $t_1, t_2, \cdots, t_q$（$0 \le t_i \le 10^9$），其中 t_i 表示第 i 个问题的时间限制。

保证所有数据 n 之和与 q 之和均不超过 2×10^5。

Output

每组数据输出一行 q 个由单个空格分隔的整数，其中第 i 个整数表示 t_i 秒后最多共能保留多少体积的水。

Example

standard input	standard output
2 4 5 4 7 6 2 1 3 2 3 3 1 2 4 19 47 21 13 5 14 2 3 5 5 2 6 1 4	4 14 8 43 67 38 77 48

Problem H. Pen Pineapple Apple Pen

给定仅由小写英文字母组成的字符串 $S = s_1 s_2 \ldots s_n$，受 PIKO 太郎爆红歌曲《Pen-Pineapple-Apple-Pen》（PPAP）的启发，您需要计算从左到右选择四个互不重叠的子串的方案数，使其模仿标志性的 PPAP 结构：

pen pineapple apple pen

更具体地，选择八个整数 $i_1, j_1, i_2, j_2, i_3, j_3, i_4, j_4$ $(1 \le i_1 \le j_1 < i_2 \le j_2 < i_3 \le j_3 < i_4 \le j_4 \le n)$，使得：

- 第一个子串与第四个子串相同：$s_{i_1} s_{i_1+1} \ldots s_{j_1} = s_{i_4} s_{i_4+1} \ldots s_{j_4}$。这对应了开头和结尾相同的 “pen”。
- 第三个子串是第二个子串的严格后缀。即存在一个整数 k_2 $(i_2 < k_2 \le j_2)$，使得 $s_{k_2} s_{k_2+1} \ldots s_{j_2} = s_{i_3} s_{i_3+1} \ldots s_{j_3}$。这体现了 “pineapple” 与 “apple” 之间的关系，其中后者是前者的严格后缀。

您需要计算满足上述条件的所有合法八元组 $(i_1, j_1, i_2, j_2, i_3, j_3, i_4, j_4)$ 的数量。

Input

每个测试文件仅有一组测试数据。

第一行输入一个字符串 S $(1 \le |S| \le 5 \times 10^3)$，表示给定的仅由小写英文字母组成的字符串。

Output

输出一行一个整数，表示满足 PPAP 风格的选择方案数。由于答案可能很大，输出答案对 998 244 353 取模的结果。

Examples

standard input	standard output
ppaap	1
ppapppap	16
penpineappleapplepen	1502

Problem I. 吃饭

Grammy 有两位队友小 A 和小 B。平时，两位队友每天都会出去吃好吃的餐厅，而 Grammy 一般都吃学校食堂。因为学校食堂省时省钱，而出去吃饭有时候还要承担打车的费用。即使如此，有时 Grammy 也会选择和队友们一起出去改善伙食。

马上就要月底了，Grammy 的生活费也只剩下 m 元，她需要用这些钱度过接下来的 n 天。在第 i 天，她有两种选择：

- 在食堂吃饭，获得 a_i 的满意度，并花费 b_i 元。
- 和队友们出去吃，获得 c_i 的满意度，并花费 d_i 元。

由于小 A 和小 B 每天都出去吃，因此每天都会产生打车费用，其中第 i 天的打车费用为 e_i。三人为此制定了一个付打车费的规则：

- 若第 i 天 Grammy 没有出去吃，则本次打车费有 $\frac{p_i}{100}$ 的概率由小 A 支付，$(1-\frac{p_i}{100})$ 的概率由小 B 支付。
- 若第 i 天 Grammy 也出去吃了，且自 Grammy 上次支付打车费以来（若 Grammy 未支付过打车费，则从第一天算起），小 A 和小 B 都各自支付了至少一次打车费（无论当时 Grammy 是否出去吃），则本次打车费由 Grammy 支付。否则按上一条规则，由一位队友支付。

Grammy 将在每天吃饭前选择今天的行动。她每天都能知道前一天打车费的支付情况，且目标是最大化总满意度的期望值，同时在最差情况下的总开销不能超过 m 元。问：如果她每一天的决策都是最优的，总满意度的期望值是多少。

Input

每个测试文件仅有一组测试数据。

第一行输入两个整数 n 和 m（$1 \le n \le 2 \times 10^3$，$1 \le m \le 5 \times 10^3$），表示天数和总开销上限。

对于接下来的 n 行，第 i 行输入六个整数 a_i，b_i，c_i，d_i，e_i 和 p_i（$0 \le b_i, d_i, e_i \le 5 \times 10^3$，$0 \le a_i, c_i \le 10^9$，$0 \le p_i \le 100$）。它们的定义如上所述。

Output

输出一行一个数，表示总满意度的最大期望值。如果不存在一个在最坏情况下开销小于等于 m 元的方案，则输出 -1。

如果您的答案的绝对误差或相对误差不超过 10^{-6}，则将被视为正确。更正式地，假设您的输出为 a，标准答案为 b，当且仅当 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$ 时，您的输出才会被接受。

Examples

standard input	standard output
2 10 6 4 15 8 10 50 5 2 13 5 8 50	20.000000000000
5 12 2 0 10 5 3 0 2 0 9 4 3 100 2 0 12 7 3 0 2 0 8 4 3 100 2 0 10 6 3 0	25.000000000000
6 20 1 2 3 10 1 69 5 1 12 2 2 23 0 4 2 3 5 41 6 1 6 9 0 84 1 1 14 9 5 98 3 2 13 2 4 43	35.771706880000
3 10 10 4 30 8 1 0 10 4 20 5 1 100 10 6 5 2 1 50	-1

Problem J. Trajan 算法

请回忆用于计算无向图中割点（articulation points，又称 cut vertices）的 *Tarjan* 算法，其伪代码如左侧所示。

Algorithm 1 The Correct Tarjan Algorithm

```
1: function TARJAN(i, d)
2:     visited[i] ← true
3:     depth[i] ← d
4:     low[i] ← d
5:     childCount ← 0
6:     isArticulation ← false
7:
8:     for each j in adj[i] do
9:         if j = parent[i] then
10:            continue
11:        end if
12:        if not visited[j] then
13:            parent[j] ← i
14:            TARJAN(j, d + 1)
15:            childCount ← childCount + 1
16:            if low[j] ≥ depth[i] then
17:                isArticulation ← true
18:            end if
19:            low[i] ← min(low[i], low[j])
20:        else
21:            low[i] ← min(low[i], depth[j])
22:        end if
23:    end for
24:    if (parent[i] ≠ null and isArticulation)
   or (parent[i] = null and childCount > 1)
   then
25:        Output i as articulation point
26:    end if
27: end function
```

Algorithm 2 The Incorrect Trajan Algorithm

```
1: function TRAJAN(i, d)
2:     visited[i] ← true
3:     depth[i] ← d
4:     low[i] ← d
5:     childCount ← 0
6:     isArticulation ← false
7:
8:     for each j in adj[i] do
9:         if j = parent[i] then
10:            continue
11:        end if
12:        if not visited[j] then
13:            parent[j] ← i
14:            TRAJAN(j, d + 1)
15:            childCount ← childCount + 1
16:            if low[j] ≥ depth[i] then
17:                isArticulation ← true
18:            end if
19:        end if
20:        low[i] ← min(low[i], low[j])
21:
22:
23:    end for
24:    if (parent[i] ≠ null and isArticulation)
   or (parent[i] = null and childCount > 1)
   then
25:        Output i as articulation point
26:    end if
27: end function
```

在伪代码中，$adj[i]$ 是包含所有与节点 i 相邻的节点的数组，*parent* 是一个元素初始值均为 null 的数组，*visited* 是一个元素初始值均为 false 的数组。

一些学生可能很难理解这个算法，特别是从第 19 行到第 22 行。他们无法理解为什么 *Tarjan* 算法需要以不同的方式计算 $low[i]$ 的值，所以他们可能会记住一个错误版本的算法。我们称这个错误版本的算法为 *Trajan* 算法，如右侧所示。注意，*Tarjan* 和 *Trajan* 算法几乎相同，除了函数名称和第 19 行到第 22 行的这 4 行以外。

给定一个包含 n 个节点和 m 条边的无向连通图，对于每个节点 u，问是否存在一个节点 v 和一个二维数组 adj（adj 必须表示给定的图），使得 TARJAN$(v, 0)$ 和 TRAJAN$(v, 0)$ 在节点 u 上产生不同的结果（也就是说，*Tarjan* 算法认为顶点 u 是一个割点，而 *Trajan* 算法认为不是，反之亦然）。

Input

有多组测试数据。输入的第一行包含一个整数 T（$1 \le T \le 10^4$）表示测试数据组数，对于每组测试数据：

第一行输入两个整数 n 和 m（$2 \le n \le 10^5$，$1 \le m \le 10^5$），表示给定图中的节点和边的数量。

对于接下来 m 行，第 i 行输入两个整数 u_i 和 v_i（$1 \le u_i, v_i \le n$），表示图中连接节点 u_i 和 v_i 的一条边。

保证给定的图是连通的，并且没有自环或重边。另外，所有数据的 n 之和以及 m 之和不超过 10^6。

Output

每组数据输出一行，包含一些由单个空格分隔的整数，并按升序排列。每个整数是 *Tarjan* 和 *Trajan* 算法可能在其上产生不同结果的节点。如果没有这样的节点，则输出一行 `Empty`。

Example

standard input	standard output
2 9 12 1 3 2 3 1 2 3 6 6 9 9 3 4 6 5 6 4 5 7 9 8 9 7 8 2 1 1 2	3 6 9 Empty

Note

对于第一组样例数据：

- 对于 $u = 3$ 考虑 $v = 1$ 和 $adj = [[3, 2], [3, 1], [1, 2, 6, 9], [6, 5], [6, 4], [4, 5, 3, 9], [9, 8], [9, 7], [7, 8, 3, 6]]$。
- 对于 $u = 6$ 考虑 $v = 4$ 和 $adj = [[2, 3], [1, 3], [9, 6, 2, 1], [6, 5], [6, 4], [4, 5, 3, 9], [9, 8], [9, 7], [7, 8, 3, 6]]$。
- 对于 $u = 9$ 考虑 $v = 7$ 和 $adj = [[2, 3], [1, 3], [9, 6, 2, 1], [5, 6], [4, 6], [9, 3, 5, 4], [9, 8], [9, 7], [7, 8, 3, 6]]$。

注意，所有的 adj 在解释中都是 1-索引的。

Problem K. 象棋

中国象棋是一项老少皆宜的热门运动，此事在第 49 届 ICPC 国际大学生程序设计竞赛世界总决赛的现场亦有记载。

摄于第 49 届 ICPC 国际大学生程序设计竞赛世界总决赛现场

今天，刚学会象棋规则的您要来挑战象棋大师秋私语。大师不愿意欺负您，所以只和您玩一种简易版的象棋，规则如下：

- 象棋的棋盘的大小为 9 列 10 行，仅放置两枚棋子：马和车。用 (x, y) 表示一枚棋子被放置在第 x 列第 y 行。车和马只能在棋盘内移动，不可以走到棋盘之外。
- 两名玩家轮流移动自己的棋子。若一名玩家移动后，他/她的棋子恰好移动到了对方棋子的位置，则称他/她”吃掉了”对方的棋子，对局结束。
- 马的移动规则如下：
  - 可以从 (x_N, y_N) 移动到 $(x_N + 2, y_N \pm 1)$，若 $(x_N + 1, y_N)$ 没有放置棋子。
  - 可以从 (x_N, y_N) 移动到 $(x_N - 2, y_N \pm 1)$，若 $(x_N - 1, y_N)$ 没有放置棋子。
  - 可以从 (x_N, y_N) 移动到 $(x_N \pm 1, y_N + 2)$，若 $(x_N, y_N + 1)$ 没有放置棋子。
  - 可以从 (x_N, y_N) 移动到 $(x_N \pm 1, y_N - 2)$，若 $(x_N, y_N - 1)$ 没有放置棋子。
- 车可以移动到同一行或同一列的任意不同位置，但不可跳过其它棋子。
- 玩家每轮必须移动自己的棋子，不能让棋子留在当前位置。可以证明，在任何情况下，当前玩家均有移动棋子的方法。

大师操纵马，您操纵车，大师先手。问：车是否存在一种策略，使得无论马如何应对，车都能在不被马吃掉的前提下，在有限轮次内吃掉马。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 8010$）表示测试数据组数，对于每组测试数据：

第一行输入四个整数 x_N，y_N，x_R，y_R（$1 \le x_N, x_R \le 9$，$1 \le y_N, y_R \le 10$），其中 (x_N, y_N) 是马的初始位置，(x_R, y_R) 是车的初始位置。保证两枚棋子的初始位置不同。

Output

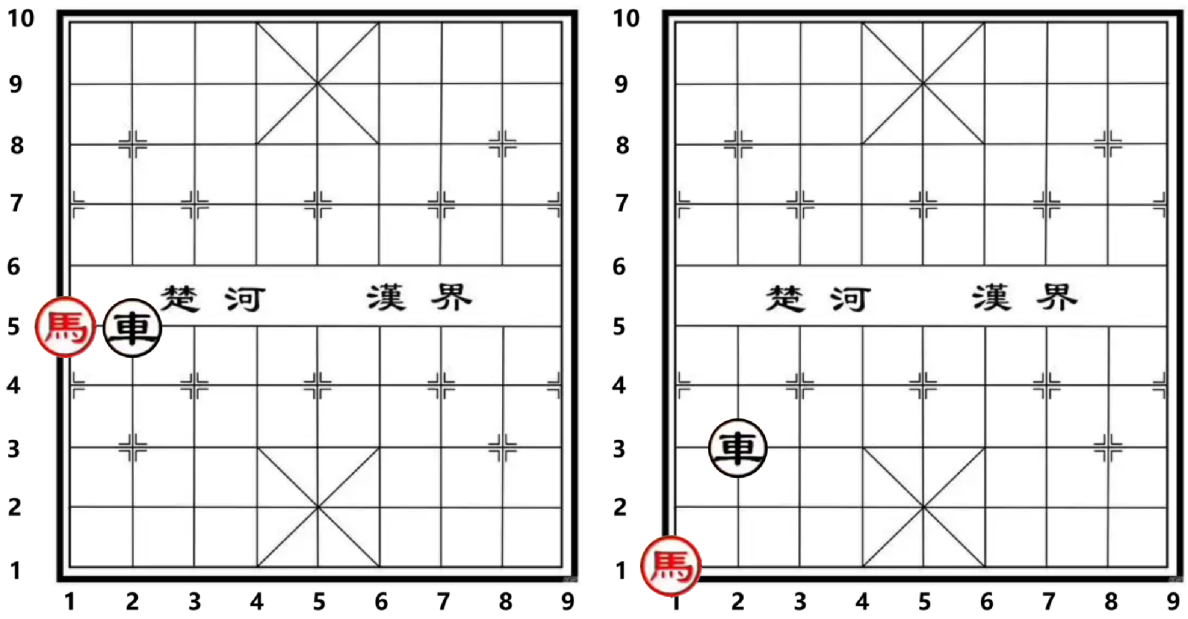
每组数据输出一行。若车存在相应策略，输出 YES，否则输出 NO。

Example

standard input	standard output
2 1 5 2 5 1 1 2 3	YES NO

Note

样例数据如下图所示。其中红色的棋子代表马，黑色的棋子代表车。



Problem L. 区域赛冠军

Arona 想赢得区域赛冠军。因此，Plana 根据关键词" 区域" 为 Arona 生成了以下问题。

给定三个整数 n、m 和 k，您需要在平面上绘制 n 个圆、m 个三角形和 k 条直线，将平面划分成尽可能多的区域。

当然，Arona 的绘画技巧和数学能力都很差。所以她来向您寻求帮助。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 5 \times 10^3$）表示测试数据组数，对于每组测试数据：

第一行输入三个整数 n，m 和 k（$0 \le n, m, k \le 100$，$n + m + k > 0$），表示圆、三角形和直线的数量。

保证所有数据 $(\max(n, m, k))^3$ 之和不超过 10^7。

Output

对于每组数据：

首先输出一行一个整数，表示可以划分出的最大区域数量。

然后输出 n 行。每行输出三个由单个空格分隔的整数 x，y 和 R（$-10^3 \le x, y \le 10^3$，$1 \le R \le 10^3$），表示圆心的坐标以及圆的半径。

接下来输出 m 行。每行输出六个由单个空格分隔的整数 x_1，y_1，x_2，y_2，x_3 和 y_3（$-10^3 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10^3$，$(x_1, y_1) \ne (x_2, y_2)$，$(x_1, y_1) \ne (x_3, y_3)$，$(x_2, y_2) \ne (x_3, y_3)$），表示三角形三个顶点的坐标。

最后输出 k 行。每行输出四个由单个空格分隔的整数 x_1，y_1，x_2 和 y_2（$-10^3 \le x_1, y_1, x_2, y_2 \le 10^3$，$(x_1, y_1) \ne (x_2, y_2)$），表示直线上的两个点的坐标。

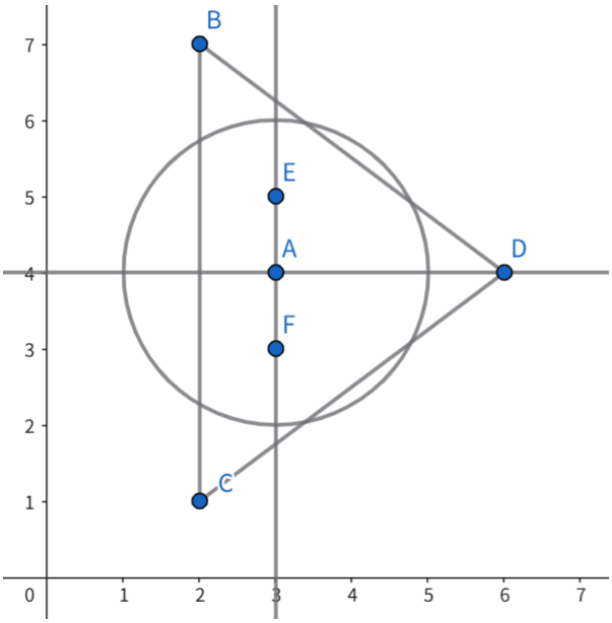
可以证明，即使在上述限制下，我们也总能达成最优解。

Example

standard input	standard output
3 1 0 1 0 1 1 1 1 2	4 0 0 1 -2 0 2 0 4 0 0 0 2 2 0 0 1 1 0 18 3 4 2 2 7 2 1 6 4 3 4 6 4 3 3 3 5

Note

第三个样例数据解释如下。



Problem M. 很多凸多边形

按逆时针顺序给定一个凸多边形的 n 个顶点 $P(0), P(1), \cdots, P(n-1)$，从这些顶点中选择 k 个顶点 $P(i_0), P(i_1), \cdots, P(i_{k-1})$（$0 \le i_0 < i_1 < \cdots < i_{k-1} < n$）的方式有 $\binom{n}{k}$ 种。将被选中的顶点按逆时针顺序连接后（即对于每个 $0 \le j < k$，将顶点 $P(i_j)$ 和 $P(i_{(j+1) \bmod k})$ 连接），可以得到一个有 k 个顶点的凸多边形。

对于每个 $3 \le k \le n$，考虑通过以上方式形成的所有具有 k 个顶点的凸多边形，求这些凸多边形面积之和的两倍。

Input

每个测试文件仅有一组测试数据。

第一行输入一个整数 n（$3 \le n \le 2 \times 10^5$），表示凸多边形的顶点数量。

对于接下来 n 行，第 i 行输入两个整数 x_i 和 y_i（$-10^9 \le x_i, y_i \le 10^9$），表示凸多边形第 i 个顶点的坐标。顶点按逆时针顺序给出，且任意三个顶点不共线。

Output

输出 $(n-2)$ 行，每行输出一个整数，其中第 i 行表示 $k = i + 2$ 的答案。

可以证明答案是一个整数。由于答案可能很大，请输出答案对 998 244 353 取模后的结果。

Example

standard input	standard output
6 0 0 1 0 2 1 2 2 1 2 0 1	36 54 30 6