2026 icpc 国际大学生程序设计竞赛
全国邀请赛（深圳）

正式赛

2026 年 4 月 11 日

香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

试题列表

A	来自陈教授的问候
B	全明星对抗赛
C	一物之差
D	城市管理
E	我要验牌
F	Astra
G	贪吃蛇
H	心灵感应
I	日历立方体
J	交叉路口
K	和与积
L	暴击
M	博物馆奇妙夜

本试题册共 13 题，21 页。
如果您的试题册缺少页面，请立即通知志愿者。

承办方

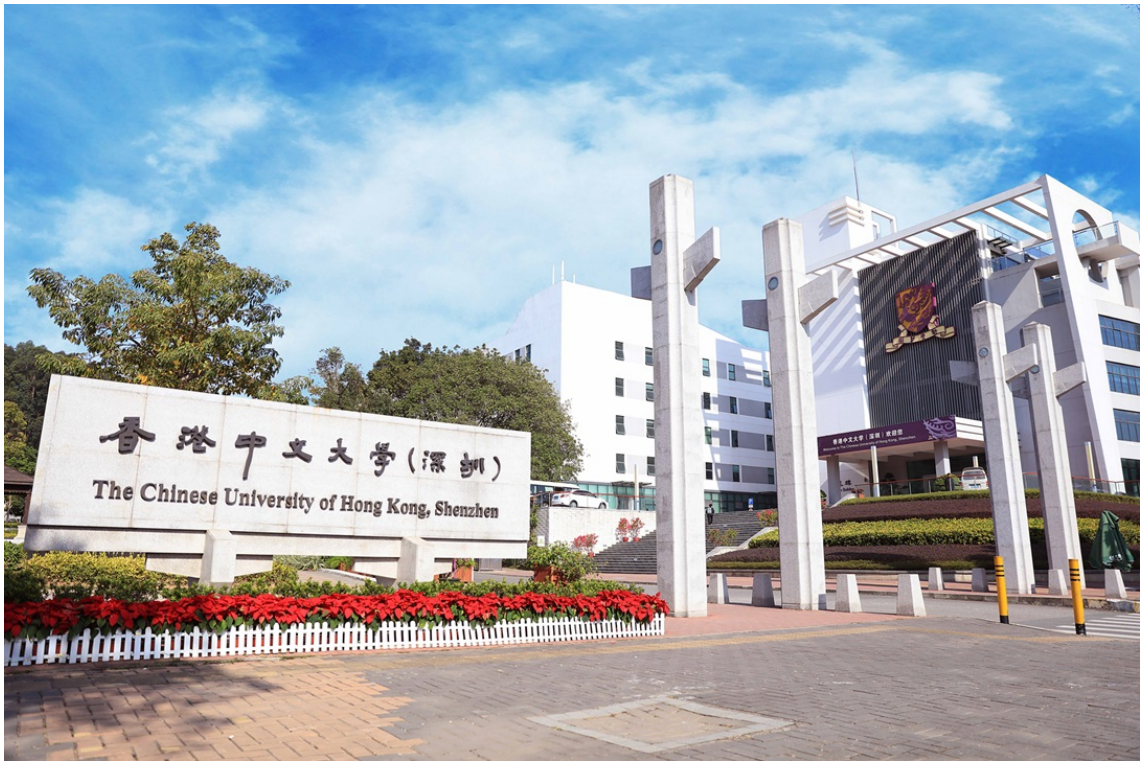
香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

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. 来自陈教授的问候

欢迎来到香港中文大学（深圳）！陈教授喜欢用仅由小写英文字母组成的字符串发送问候。



给定一个整数 n，构造一个仅由小写英文字母组成的字符串，使字符的 ASCII 码之和恰好等于 n。

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：

仅有的一行输入一个整数 n（$1 \le n \le 10^7$）。

保证所有测试数据中 n 的总和不超过 10^7。

Output

对于每组测试数据：

- 如果可以构造出一个仅由小写英文字母组成、且其 ASCII 码之和等于 n 的字符串，则首先在一行中输出 `Yes`，然后在第二行输出这样一个字符串。
- 否则，如果无法构造这样的字符串，则仅在一行中输出 `No`。

Example

standard input	standard output
3 2257 2269 96	Yes greetingsfromprofchen Yes welcometocuhkshenzhen No

Note

以下是小写英文字母的 ASCII 码对照表：

字母	ASCII	字母	ASCII	字母	ASCII
a	97	j	106	s	115
b	98	k	107	t	116
c	99	l	108	u	117
d	100	m	109	v	118
e	101	n	110	w	119
f	102	o	111	x	120
g	103	p	112	y	121
h	104	q	113	z	122
i	105	r	114		

Problem B. 全明星对抗赛

今年的大奖赛将打破传统，举办一场特别的全明星对抗赛：两支精英队伍，红队与蓝队，将在一场一对一的编程对决中正面交锋。与标准赛制不同，本次表演赛允许队伍人数灵活调整，只要确保每位选手恰好被分配到其中一队即可。

共有 n 位从名人堂中选出的全明星选手。每位选手 i 在三个核心维度上接受评估：算法洞察力、实现速度和调试韧性。这些能力被量化为正整数 (x_i, y_i, z_i)，构成一个三维能力向量，用以刻画其竞赛风格。此外，每位选手 i 还被赋予一个正整数协同价值 w_i，反映其对团队整体表现的放大作用。一支队伍的总价值定义为其所有成员协同价值的乘积。

为维护比赛公平性并防止风格高度相似的选手扎堆，组委会制定了严格的多样性规则：在同一支队伍中，任意两名选手的能力向量之间的欧几里得距离必须至少为 d。形式化地，对于同一队中的任意两名不同选手 i 和 j，必须满足 $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \geq d$。这一规则旨在确保每支队伍都包含多样化的解题思路。

组委会希望全面分析所有合法分配方案的组合结构。具体地，对于每个整数 $k = 1, 2, \ldots, n-1$，他们要求计算在所有合法分组中，红队恰好包含 k 名选手时，红队价值的总和。

由于答案可能很大，请将答案对 998 244 353 取模后输出。

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 d（$2 \leq n \leq 10^5$，$1 \leq d \leq 10^8$）。

对于接下来的 n 行，第 i 行输入四个整数 x_i，y_i，z_i 和 w_i（$1 \leq x_i, y_i, z_i, w_i \leq 10^8$），表示第 i 位选手的能力向量和协同价值。

Output

输出 $(n-1)$ 行，其中第 i 行输出一个整数，表示在所有合法分组中，红队恰好包含 i 名选手时，红队价值的总和对 998 244 353 取模后的结果。

Examples

standard input	standard output
3 1 1 1 1 1 1 2 1 10 1 2 3 100	111 1110
3 2 1 1 1 1 1 2 1 10 1 2 3 100	11 1100
3 2 2 1 1 1 1 2 1 10 1 1 2 100	0 0

Problem C. 一物之差

这是一道交互题。请注意在每次输出后刷新输出缓冲区。您可以使用以下方式刷新输出：

- 在 C/C++ 中使用 `fflush(stdout)` 或 `cout.flush()`；
- 在 Java 和 Kotlin 中使用 `System.out.flush()`；
- 在 Python 中使用 `sys.stdout.flush()`。

有 n 个人和 m 件物品。每件物品必须被分配给恰好一个人。

不同的人对物品集合的满意度可能不同。对于任意一个物品集合，一个人可以评估自己对该集合的满意度。您并不知道这些满意度函数的具体形式，但可以通过询问来获取它们的值。您最多可以进行 $n \times m$ 次询问。

更正式地，对每个人 i，存在一个函数 $u_i(S)$，它为任意物品子集 S 赋予一个非负整数值。该函数满足以下性质：

- $u_i(\varnothing) = 0$；
- 若 $S \subseteq T$，则 $u_i(S) \le u_i(T)$。

换句话说，给一个人更多的物品不会降低其满意度。

然而，人们可能会比较敏感。如果第 i 个人发现另一个人 j 拥有的物品集合严格优于自己的，此人可能会抱怨。

幸运的是，人们也具有一定的宽容性：只要从 j 的物品集合中移除某一件物品后，这种优势就不再存在，那么就不会产生抱怨。

更正式地，设 s_i 表示分配给第 i 个人的物品集合。您的目标是将所有物品分配给各个人，使得对任意两个人 i 和 j，都满足以下条件：如果第 i 个人认为第 j 个人的物品集合严格优于自己的（即 $u_i(s_j) > u_i(s_i)$），则存在某个物品 $x \in s_j$，使得 $u_i(s_j \setminus \{x\}) \le u_i(s_i)$。

Input

每个测试文件中只有一组测试数据。

第一行输入两个整数 n 和 m（$1 \le n \le 100$，$1 \le m \le 500$），分别表示人数和物品数量。

Interaction Protocol

您可以通过询问来获取满意度值。

若要询问第 i 个人对物品子集 $\{x_1, x_2, \ldots, x_k\}$ 给出的满意度值，输出一行：

$$\texttt{?}\ i\ k\ x_1\ x_2\ \ldots\ x_k$$

其中 $1 \le i \le n$，$0 \le k \le m$，且 $x_1, x_2, \ldots, x_k$ 是两两不同的整数，满足 $1 \le x_t \le m$。

在刷新输出缓冲区后，您的程序需要读入一个整数 v（$0 \le v \le 10^9$），其值等于 $u_i(\{x_1, x_2, \ldots, x_k\})$。

您最多可以进行 $n \times m$ 次询问。另外需要注意，交互器是自适应的。具体而言，询问的返回结果可能依赖于您之前的询问，但始终与某组满足题面所有约束的函数 $u_i(\cdot)$ 一致。

当您准备好分配物品时，输出一行：

$$\texttt{!}\ a_1\ a_2\ \ldots\ a_m$$

其中 a_j 表示物品 j 被分配给的人（$1 \le a_j \le n$）。分配物品不算一次询问。

在刷新输出缓冲区后，您的程序应立即终止。

Example

standard input	standard output
2 3 10 20 10 5	? 1 1 1 ? 1 2 2 3 ? 2 1 1 ? 2 2 2 3 ! 2 1 1

Problem D. 城市管理

您正在管理一座城市。城市中将会建立一些工厂，同时也会有一些工人来找工作。您需要对这些事务进行合理的安排，以追求最大的收益。

每座工厂有一个产能值 x_i，表示在一轮生产中：

- 若有一名工人在里面工作，则会产生 x_i 的收益。
- 若没有工人在里面工作，则不会产生收益。
- 每座工厂中最多有一名工人工作。

您已经预知，在将来的 n 天，每天会发生如下两种事件之一：

- 有一名工人来找工作。这位工人到来后，您可以重新分配所有工人（包括已经在工厂里工作的工人）的岗位。具体来说，对于每名工人，您可以指定他/她到某座工厂工作，或者让他/她待岗。
- 有一座产能值为 x_i 的工厂被建立。这座工厂被建立后，如果有待岗的工人，您可以选择是否让一名待岗工人去该工厂工作。当然，您也可以让工厂暂时空置。特别地，如果没有待岗工人，那么这座工厂就只能空置了。

当天的事件执行完毕后，所有工厂都会进行一轮生产，并根据前文所述规则产生收益。

您需要求出最大总收益。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 10^6$），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（$1 \le n \le 2 \times 10^5$），表示天数。

对于接下来的 n 行，第 i 行首先输入一个字符 op_i（$op_i \in \{\texttt{W}, \texttt{F}\}$），表示第 i 天的事件类型。若 $op_i = \texttt{W}$，那天会有一名工人来找工作；否则若 $op_i = \texttt{F}$，则再输入另一个整数 x_i（$1 \le x_i \le 10^7$），表示那天有一座产能值为 x_i 的工厂被建立。

保证所有测试数据中 n 的总和不超过 10^6。

Output

每组测试数据输出一行，包含一个整数，表示最大总收益。

Example

standard input	standard output
2 8 F 1 F 2 W F 100 W W F 10000 W 9 W F 51 F 1 F 1 F 1 F 1 F 100 F 100 W	20509 557

Note

第一组样例数据解释如下：

天数	操作	当天收益
1	/	0
2	/	0
3	让一名工人待岗	0
4	让一名工人去新工厂工作	100
5	重新分配工人到工厂 2 和 100	102
6	重新分配工人到工厂 2 和 100，让一名工人待岗	102
7	让一名工人去新工厂工作	10102
8	重新分配工人到所有工厂	10103

总收益为 $0+0+0+100+102+102+10102+10103=20509$。

Problem E. 我要验牌

您需要解决一个关于扑克游戏《斗地主》的问题。

游戏简介

《斗地主》是一种三人轮流出牌的游戏。设三名玩家分别为 A，B，C。每名玩家都有若干张手牌，游戏不断进行，直到有一名玩家将手牌出完。

玩家 A 和 B 属于农民阵营。若 A 或 B 中任意一人将手牌出完，则农民获胜。玩家 C 单独组成地主阵营，若 C 将手牌出完，则地主获胜。

出牌规则

本题中，每张牌上都写有一个 $[1, n]$ 中的整数，称为该牌的点数。点数从小到大依次为 $1, 2, \ldots, n$。

本题只允许出现以下两种牌型：

- 单张：任意一张牌。
- 对子：点数相同的两张牌。

与常规的斗地主游戏不同，本题中单张只能在手中仅有一张该点数的牌时打出。换句话说，手牌中的对子不能拆成两个单张分开打出。

在每一轮出牌中，会有一名玩家作为该轮的起始玩家。起始玩家首先行动，并且可以打出任意一个合法牌型。随后，从下一位玩家开始，玩家们依次行动。每名玩家在自己的回合中可以进行以下两种操作之一：

- 打出点数严格更大的牌，且需要与本轮最后一次打出的牌型相同；
- 选择跳过。

当某位玩家出牌后，如果其余两名玩家都依次选择跳过，则当前这一轮出牌结束。最后一次成功出牌的玩家成为下一轮的起始玩家，并开始新的一轮。

问题

本题中，您需要考虑如下局面：

- 地主 C 只剩下一张牌，其点数为 p_c。
- 两位农民 A 和 B 的手牌分别由两个长度为 n 的字符串 s_a 和 s_b 描述。字符串中仅包含字符 `0`、`1`、`2` 。其中，第 i 个字符表示对应玩家拥有多少张点数为 i 的牌。

所有玩家的手牌均为公开信息，即每位玩家在决策时均知晓所有人的确切手牌。玩家 A 是第一轮出牌的起始玩家，三名玩家始终按照 $\mathrm{A} \to \mathrm{B} \to \mathrm{C} \to \mathrm{A} \to \cdots$ 的顺序轮流行动。您需要判断在所有玩家都采取最优策略的情况下，农民阵营能否获胜。

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（$1 \le n \le 2.5 \times 10^5$）。

第二行输入一个长度为 n 的字符串 s_a，仅包含 `0`、`1`、`2`，表示农民 A 的手牌。

第三行输入一个长度为 n 的字符串 s_b，仅包含 `0`、`1`、`2`，表示农民 B 的手牌。

第四行输入一个整数 p_c（$1 \le p_c \le n$），表示地主 C 唯一的一张牌的点数。

保证所有玩家都至少有一张牌。

保证所有测试数据中 n 的总和不超过 10^6。

Output

每组测试数据输出一行。如果农民阵营可以获胜，输出 `Yes`；否则输出 `No`。

Example

standard input	standard output
4 9 001110201 002110211 7 9 110200000 222000000 7 9 222000000 110000000 7 9 111000002 111000210 7	Yes No Yes No

Note

以下对第一组样例数据进行解释。

#	A	B	C	出牌
1	3,4,5,7,7,9	3,3,4,5,7,7,8,9	7	A (3) → B (8) → C (跳过) → A (9) → B (跳过) → C (跳过)
2	4,5,7,7	3,3,4,5,7,7,9	7	A (4) → B (9) → C (跳过) → A (跳过)
3	5,7,7	3,3,4,5,7,7	7	B (3,3) → C (跳过) → A (7,7) → B (跳过) → C (跳过)
4	5	4,5,7,7	7	A (5)

Problem F. Astra

《Astra》是一款画风优美的桌游。游戏中，玩家通过轮流标记星星来发现星座。该游戏将规划与行动巧妙结合，规则简洁直观，游玩起来易于上手且节奏紧凑。

BoardGameGeek 用户 @kavics004 上传的照片

考虑一场由 Alice 和 Bob 进行的二人游戏。游戏中有一个树形星座，可以看作由 n 颗星星（编号从 1 到 n）和 $(n-1)$ 条边组成的连通图。一开始，只有一颗星星 s 被标记。Alice 和 Bob 轮流根据以下规则标记剩余的星星：

- 玩家每轮必须标记至少 1 颗，至多 k 颗星星。
- 每颗星星只能被标记一次。
- 设 $a_1, a_2, \cdots, a_p$ 是在一轮中依次被标记的星星：
  - 标记 a_1 时，它必须与一颗已被标记的星星相邻。
  - 对于所有 $2 \le i \le p$，标记 a_i 时，它必须是 a_{i-1} 的邻居。

称两颗星星 u 和 v 是邻居，若它们被一条边直接相连。

标记最后一颗星星的玩家将赢得该星座。已知 Alice 先手，且两位玩家都以最佳策略进行游戏，对于每个 $1 \le s \le n$，确定谁将赢得星座。

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 k（$2 \le n \le 2 \times 10^3$，$1 \le k < n$），表示星星总数和每轮最多标记的星星数量。

对于接下来的 $(n-1)$ 行，第 i 行输入两个整数 u_i 和 v_i（$1 \le u_i, v_i \le n$），表示有一条边连接星星 u_i 和 v_i。

保证所有测试数据中 n 的总和不超过 2×10^3。

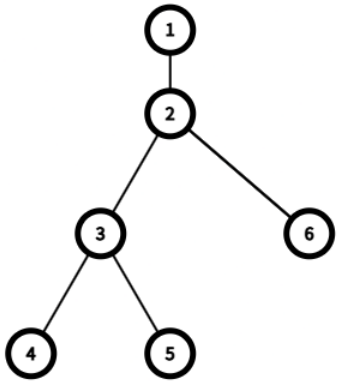
Output

每组测试数据输出一行，包含一个字符串 $d_1d_2\cdots d_n$ $(d_i \in \{0,1\})$，其中 d_i 表示当 $s=i$ 时谁将赢得星座。如果 Alice 获胜，则 $d_i=1$；否则如果 Bob 获胜，则 $d_i=0$。

Example

standard input	standard output
3 6 2 1 2 3 2 3 4 5 3 2 6 3 1 1 2 2 3 3 2 1 2 2 3	011000 000 101

Note



第一组样例数据如上图所示。

- 考虑 $s=1$。在第一轮中，Alice 有三个选择：标记 $\{2\}$，$\{2,3\}$ 或 $\{2,6\}$。如果 Alice 选择 $\{2,6\}$，Bob 可以标记 $\{3\}$，这样就会剩下 2 颗独立的星星。在接下来的回合中，每位玩家只能标记一颗星星，Bob 将成为赢家。Alice 的其他选择也可以进行类似的分析。
- 考虑 $s=2$。在第一轮中，Alice 有五个选择：标记 $\{1\}$，$\{3\}$，$\{6\}$，$\{3,4\}$ 或 $\{3,5\}$。Alice 可以选择标记 $\{3\}$，这样就会剩下 4 颗独立的星星。在接下来的回合中，每位玩家只能标记一颗星星，Alice 将成为赢家。

对于第二组样例数据，由于 $k=1$，将会恰好进行 $3-1=2$ 轮，因此 Bob 总会获胜。

对于第三组样例数据，如果 $s=1$ 或 $s=3$，Alice 可以直接标记剩下的 2 颗星星并赢得该星座。

Problem G. 贪吃蛇

这是一道交互题。请注意在每次输出后刷新输出缓冲区。您可以使用以下方式刷新输出：

- 在 C/C++ 中使用 `fflush(stdout)` 或 `cout.flush()`；
- 在 Java 和 Kotlin 中使用 `System.out.flush()`；
- 在 Python 中使用 `sys.stdout.flush()`。

您正在一个 $n \times m$ 的网格上玩经典的贪吃蛇游戏。行从上到下编号为 1 到 n，列从左到右编号为 1 到 m。我们用 (i, j) 表示第 i 行第 j 列的格子。

蛇可以用一系列坐标对 $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ 表示其身体所在的位置，其中 k 表示蛇的长度。蛇头位于 (x_1, y_1)，蛇尾位于 (x_k, y_k)，且相邻的身体部分位于共享一条边的格子中。

初始时，蛇的长度为 1，位于给定的格子 (r_s, c_s)。

有 4 种指令可以移动蛇：

- `U`：让蛇向上移动一步。蛇头将移动到 $(x_1 - 1, y_1)$。
- `D`：让蛇向下移动一步。蛇头将移动到 $(x_1 + 1, y_1)$。
- `L`：让蛇向左移动一步。蛇头将移动到 $(x_1, y_1 - 1)$。
- `R`：让蛇向右移动一步。蛇头将移动到 $(x_1, y_1 + 1)$。

当蛇头移动时，身体的每个部分也会相应移动。具体地，身体的第 i 部分 $(2 \le i \le k)$ 会移动到第 $(i-1)$ 部分在指令执行前所在的位置。同时，蛇尾原先占据的格子变为空格。

蛇不能移出网格。此外，蛇不能与自身碰撞——您必须保证在任何指令执行后，身体的任意两部分不会占据同一个格子。考虑如下边界情况：蛇头位于 (x_1, y_1)，蛇尾位于 (x_k, y_k)。如果蛇头接下来要移动到 (x_1', y_1')，则允许 $(x_1', y_1') = (x_k, y_k)$：我们可以想象一个现实场景中，蛇头移入该格子的同时蛇尾恰好移出。类似地，当 $k = 2$ 时，允许通过一条指令交换蛇头和蛇尾的位置。

有 $(nm - 1)$ 个苹果会依次出现。每次，一个苹果出现在某个当前未被蛇占据的格子 (r, c) 上。您需要输出一串指令，将蛇头移动到苹果所在的位置。蛇头必须恰好在最后一条指令执行后到达 (r, c)，并且在此之前不能进入 (r, c)。

对于您输出的每串指令，除最后一条外都是普通移动：蛇尾如上所述腾出其所在格子。最后一条指令是进食移动：当蛇头到达 (r, c) 时，蛇吃掉苹果。在这次移动中，蛇尾不会腾出其所在格子，因此蛇的长度增加 1。

您的任务是成功吃掉所有 $(nm - 1)$ 个苹果。

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：

第一行输入四个整数 n，m，r_s 和 c_s $(2 \le n, m \le 50,\ nm \le 100,\ 1 \le r_s \le n,\ 1 \le c_s \le m)$，表示网格的大小和蛇的初始位置。

Interaction Protocol

您需要依次吃掉所有 $(nm - 1)$ 个苹果。对于每个苹果：

- 首先读入一行两个整数 r 和 c $(1 \le r \le n,\ 1 \le c \le m)$，表示苹果的位置。保证苹果的位置当前未被蛇占据。

- 然后输出一个由字符 U、D、L、R 组成的字符串——将蛇头从当前位置移动到 (r, c) 的指令序列。为了增加难度，我们限制字符串的长度不超过 nm。

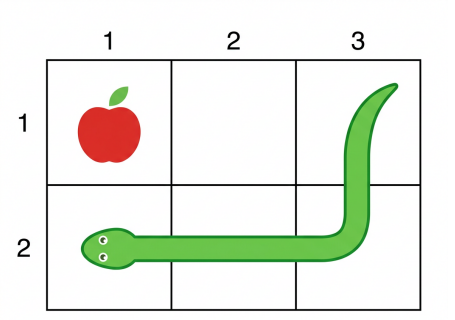
每输出一行后，不要忘了刷新输出缓冲区。可以证明答案总是存在的。

Example

standard input	standard output
2 2 3 1 2 2 1 1 3 2 1 1 1 1 2 2 2 1 1 2 2 1 1 1 2	 LD URR DLL U R RD LU R

Note

下图展示了第一组样例数据吃掉 3 个苹果后的情况。



Problem H. 心灵感应

这是一道通信题。

Alice 和 Bob 正在为 Christina 表演一个心灵感应魔术。Christina 选择一个子集 $S \subset \{1, 2, \ldots, n\}$ 满足 $|S| < \frac{n}{2}$。然后，Alice 向这个子集添加一个整数 x，使得 $x \in \{1, 2, \ldots, n\}$ 且 $x \notin S$。Bob 只能获得 Alice 添加后的结果集合，并需要推断出 Alice 添加的整数。

Interaction Protocol

您的程序在每组测试中将被运行两次。在第一次运行中，您的程序扮演 Alice。在第二次运行中，您的程序扮演 Bob。

每次运行包含多组测试数据。

第一次运行

第一行输入字符串 `Alice`。第二行输入一个整数 T（$1 \le T \le 5 \times 10^5$），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 k（$3 \le n \le 10^6$，$1 \le k < \frac{n}{2}$）。第二行输入 k 个互不相同的整数 $a_1, a_2, \ldots, a_k$（$1 \le a_i \le n$），表示 Christina 选择的子集。

对于每组测试数据，您的程序应输出一行，包含 $(k+1)$ 个互不相同的整数 $b_1, b_2, \ldots, b_{k+1}$（$1 \le b_i \le n$），使得 $\{a_1, a_2, \ldots, a_k\} \subset \{b_1, b_2, \ldots, b_{k+1}\}$。

第二次运行

您的程序将被重启以进行第二次运行。

第一行输入字符串 `Bob`。第二行输入一个整数 T（$1 \le T \le 5 \times 10^5$），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 k（$3 \le n \le 10^6$，$1 \le k < \frac{n}{2}$）。第二行输入 $(k+1)$ 个互不相同的整数 $b_1, b_2, \ldots, b_{k+1}$（$1 \le b_i \le n$），这正是您在第一次运行中针对该测试数据输出的集合（顺序可能不同）。

对于每组测试数据，您的程序应输出一行，包含一个整数，表示您在第一次运行中添加的整数。

请注意：第二次运行的测试数据顺序可能和第一次运行时不同。只有您对所有测试数据都正确推断出被添加的整数，您的答案才被视为正确。

保证所有测试数据中 n 的总和不超过 3×10^6。

Examples

standard input	standard output
Alice 2 7 3 3 1 5 5 1 2	5 6 1 3 3 2
Bob 2 5 1 3 2 7 3 3 5 1 6	3 6

Note

示例展示了某个解决方案在样例数据上的两次运行。

第一次运行中，对于第一组样例数据，Christina 从 $\{1, 2, \ldots, 7\}$ 中选择了子集 $S = \{1, 3, 5\}$（以顺序 `3 1 5` 给出），满足 $|S| = 3 < \frac{7}{2}$。Alice 需要添加一个不在 S 中的整数，并输出大小为 4 的结果集合。她选择添加整数 6，因此输出集合 $\{1, 3, 5, 6\}$（以顺序 `5 6 1 3` 给出）。

第二次运行中，第二组样例数据对应第一次运行中的第一组样例数据。Bob 收到 $(n, k) = (7, 3)$ 和集合 $\{1, 3, 5, 6\}$（以顺序 `3 5 1 6` 给出）。借助 Alice 与 Bob 事先约定的策略，他能够推断出被添加的整数是 6，并输出 `6`。

Problem I. 日历立方体

Colin 正在设计一种使用两个立方体的特殊日历。

每个立方体恰好有 6 个面，并且每个面上标有一个从 0 到 8 的数字。Colin 希望用这两个立方体表示两位数的日期。

例如，上图展示了两个立方体在正面显示数字 16。其中一个立方体显示数字 1，另一个立方体显示数字 6。

为了表示一个以两位形式表示的整数 $d \in [01, 99]$，需要选一个立方体显示十位数字（对于 01 至 09，十位数字为 0），另一个立方体显示个位数字。注意：数字 6 也可以表示为 9。也就是说，如果一个立方体上有 6，则它可以被用作 6 或 9。

给定两个立方体，定义它们的 MEX 为最小的无法以两位数形式表示的**正整数**。可以证明不存在两个立方体能表示从 01 到 99 的所有整数（含两端）。

给定一个整数 x（以两位数的形式给出），您需要构造两个立方体，使得它们的 MEX 恰好为 x。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 99$），表示测试数据组数。对于每组测试数据：

仅有的一行输入一个整数 x ($01 \le x \le 99$)，以两位数的形式给出。

Output

对于每组测试数据：

- 如果可以构造两个立方体，使得它们的 MEX 恰好为 x，首先输出一行 `Yes`。然后在第二行输出 12 个由单个空格分隔的整数（从 0 到 8），表示这两个立方体。前 6 个整数是第一个立方体上的数字，而最后 6 个整数是第二个立方体上的数字。
- 如果不能做到，则只要输出一行 `No`。

Example

standard input	standard output
4 01 02 99 11	Yes 0 0 0 0 0 0 0 0 0 0 0 0 Yes 1 1 1 1 1 1 0 0 0 0 0 0 No Yes 4 0 5 7 6 8 0 2 3 1 0 0

Problem J. 交叉路口

给定 n 条竖线，其中第 i 条为 $x = a_i$，另外给定 m 条横线，其中第 i 条为 $y = b_i$。沿着第 i 条竖线行进 1 单位距离需要 t_i 的时间，沿着任意一条横线行进 1 单位距离需要 t_0 的时间。

令 $f(i, j)$ 表示仅沿竖线和横线行进，从 $(0, 0)$ 移动到 (a_i, b_j) 所需的最短时间。您需要回答 q 次询问，每次询问给定四个整数 l，r，d，u，您需要计算

$$\sum_{i=l}^{r} \sum_{j=d}^{u} f(i, j)$$

由于答案可能很大，请将答案对 998 244 353 取模后输出。

Input

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

第一行输入三个整数 n，m，q（$1 \le n, m, q \le 2 \times 10^5$），表示竖线数量、横线数量和询问次数。

第二行输入 n 个整数 $a_1, a_2, \ldots, a_n$（$0 = a_1 < a_2 < \cdots < a_n \le 10^9$），表示每条竖线的 x 坐标。

第三行输入 m 个整数 $b_1, b_2, \ldots, b_m$（$0 = b_1 < b_2 < \cdots < b_m \le 10^9$），表示每条横线的 y 坐标。

第四行输入 n 个整数 $t_1, t_2, \ldots, t_n$（$1 \le t_i \le 10^9$），其中 t_i 表示沿着第 i 条竖线行进 1 单位距离需要的时间。

第五行输入一个整数 t_0（$1 \le t_0 \le 10^9$），表示沿着每条横线行进 1 单位距离需要的时间。

接下来 q 行，第 i 行包含四个整数 l_i，r_i，d_i，u_i（$1 \le l_i \le r_i \le n$，$1 \le d_i \le u_i \le m$），表示第 i 次询问。

保证所有测试数据中 n 的总和、m 的总和、以及 q 的总和均不超过 2×10^5。

Output

每组测试数据输出一行，包含 q 个整数，其中第 i 个整数表示第 i 次询问的答案对 998 244 353 取模后的结果。

Example

standard input	standard output
1 3 2 3 0 2 10 0 3 100 2 1 1 1 3 1 2 2 3 2 2 1 1 1 1	43 21 0

Problem K. 和与积

设 $f(n,k)$ 表示满足以下条件的整数对 (a,b) 的数量：

- $0 \le a,b < n$。
- $k(a+b) \equiv ab \pmod{n}$。

给定两个整数 n 和 m，求

$$\sum_{k=0}^{m} f(n,k)$$

由于答案可能很大，请将答案对 998 244 353 取模后输出。

Input

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

仅有的一行输入两个整数 n 和 m（$1 \le n \le 10^{14}$，$0 \le m < n$）。

保证最多有 10 组测试数据满足 n 大于 10^8。

Output

每组测试数据输出一行，包含一个整数，表示答案对 998 244 353 取模的结果。

Example

standard input	standard output
4 7 2 100 15 202604 11 1145141 919810	25 2500 3444208 165721266

Problem L. 暴击

在一款热门动作游戏中，商店里有 n 种不同的暴击装备可供选择。第 i 件装备有 $\frac{p_i}{100}$ 的概率触发暴击，暴击倍率为 v_i，并需要花费 w_i 枚金币购买。每件装备只能购买一次。

当多件装备同时触发暴击时，只有最高的暴击倍率有效。如果没有装备触发暴击，则有效倍率视为 0。

您最多可以花费 m 枚金币购买装备。您需要最大化所购买装备的期望有效倍率。

Input

有多组测试数据。第一行输入一个整数 T（$1 \le T \le 300$），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（$1 \le n, m \le 3 \times 10^3$），表示装备的数量和您可以花费的最大金币数。

对于接下来的 n 行，第 i 行输入三个整数 p_i，v_i 和 w_i（$1 \le p_i \le 100$，$1 \le v_i \le 10^9$，$1 \le w_i \le 3 \times 10^3$），表示第 i 件装备。

保证所有测试数据中 n 的总和以及 m 的总和均不超过 3×10^3。

Output

每组测试数据输出一行，包含一个实数，表示最大的期望有效倍率。

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 3 15 75 50 5 50 100 10 25 200 15 2 10 50 10 100 70 30 1000	68.750000000000 0.000000000000

Problem M. 博物馆奇妙夜

一名保安在博物馆内巡逻以监控展品。巡逻路线是一条由 n 个点组成的闭合折线，这些点从 1 到 n 标号，顺次连接（最后一个点连接回第一个点）形成 n 条首尾相接的线段。保安沿 $1 \to 2 \to \cdots \to n \to 1 \to \cdots$ 的顺序巡逻。当保安不在线段端点时，其视野是半径为 r、圆心角为 $2a$ 的扇形，且视野中线方向与当前巡逻线段的前进方向相同。

称从巡逻线段上的一个点 P（不含线段端点）可以看到位于点 Q 的展品，当且仅当 Q 位于以 P 为顶点的扇形区域内。给定由 n 个点组成的闭合折线巡逻路线和 m 件展品的位置，对于每个 $k = 0, 1, \ldots, m$，求能够看到恰好 k 件展品的巡逻路线的总长度。

Input

每个测试文件中只有一组测试数据。

第一行输入四个整数 n，m，r 和 a（$3 \le n \le 2 \times 10^3$，$1 \le m \le 2 \times 10^3$，$1 \le r \le 10^4$，$0 < a < 90$），分别表示闭合折线的点数，展品的数量，视野半径和视野圆心角的一半（以度为单位）。

对于接下来的 n 行，第 i 行输入两个整数 x_i 和 y_i（$-10^4 \le x_i, y_i \le 10^4$），表示折线上第 i 个点的坐标。连续的点形成巡逻线段，最后一个点连接回第一个点形成闭环。保证没有两个连续的点（包括最后一个和第一个点）是相同的。

对于接下来的 m 行，第 i 行输入两个整数 x'_i 和 y'_i（$-10^4 \le x'_i, y'_i \le 10^4$），表示第 i 件展品的坐标。

Output

输出 $(m+1)$ 行，其中第 i 行包含一个实数，表示能够看到恰好 $(i-1)$ 件展品的巡逻路线的总长度。

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
3 2 3 45 1 1 5 5 8 1 2 3 4 2	14.178145584873 2.247170915928 1.231537748691
4 1 3 60 0 0 6 6 6 0 0 6 2 1	25.279547784093 3.691014964384

Note

第一组样例数据如下图所示。A 和 B 是两件展品，G 是保安。