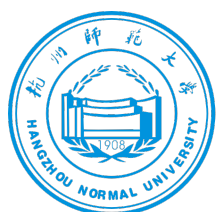


杭州师范大学第三届程序设计新生赛

计算机科技协会

December, 13, 2020

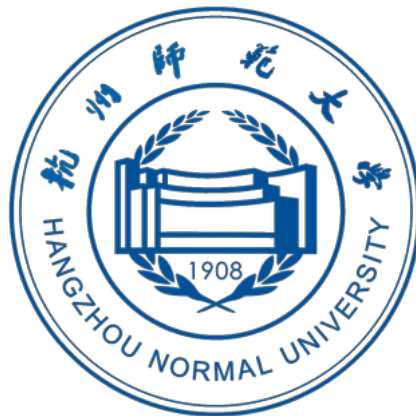


杭州师范大学
HANGZHOU NORMAL UNIVERSITY

Problems

A	XCPCIO Board CLI Easy
B	BM 算日期
C	BBpigeon Counting Trees
D	Palindrome Hard Problem
E	BM 充饥
F	kita 买礼物
G	New Game
H	Hsueh- Draw Progress
I	BM 旅游
J	大扫除
K	Kwords Find Kth Element
L	咆哮
M	XCPCIO Board CLI Hard

Do not open before the contest has started.



Problem A. XCPCIO Board CLI Easy

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 256 megabytes

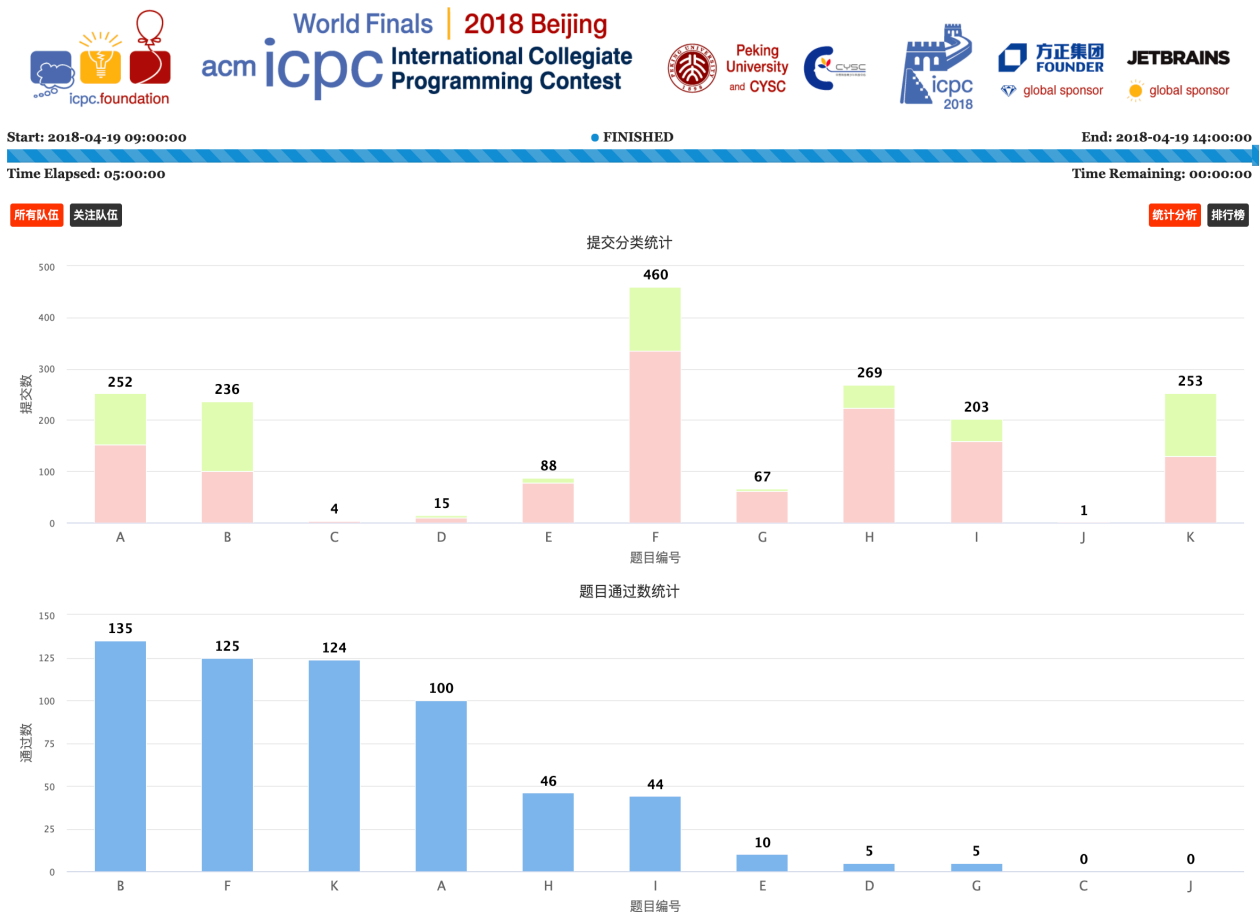
本题和 *hard version* 的区别仅仅在于队伍总数 n 的范围不同，其它均一致。

没有榜单的 *CPC 比赛怎么能称之为比赛？

Hsueh-

众所周知，在 *CPC 比赛中，榜单是一个很重要的部分。

最近，在 ICPC 和 CCPC 比赛的举办期间，有一个公益组织在 GitHub 上创建了一个项目，用于渲染 *CPC 的榜单，致力于提供更丰富的关于榜单的数据统计分析。



除了数据统计分析，XCPCIO Board 还提供了如下特性：

- 通过拖动进度条来复现某一时刻的榜单状态。
- 点开每支队伍能够显示该队伍的排名关于时间的变化情况。
- ...

Start: 2018-04-19 09:00:00

FINISHED

End: 2018-04-19 14:00:00

Time Elapsed: 05:00:00

Gold Silver Bronze Honorable First to solve problem Solved problem Attempted problem Pending judgement

Time Remaining: 00:00:00

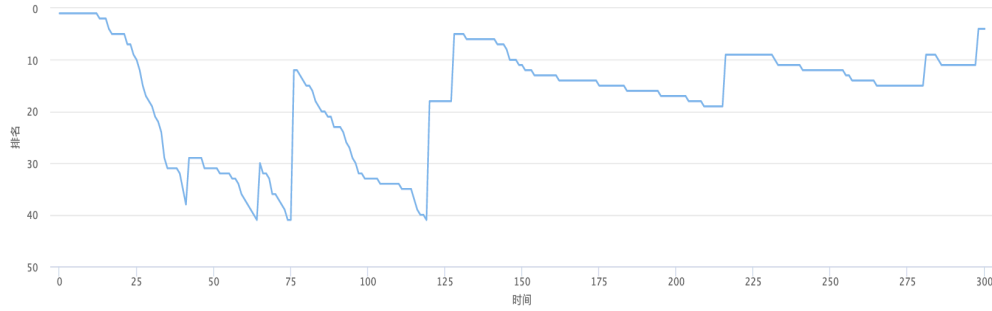
所有队伍 关注队伍

统计分析 排行榜

Place	Badge	Team	Solved	Time	A 100	B 135	C 0	D 5	E 10	F 125	G 5	H 46	I 44	J 0	K 124	Dirt
1		Moscow State University	9	1427	+	+		+	+	+	+	+	+		+	52%
2		Moscow Institute of Physics & Technology	8	960	+	+		+	+	+	+	+	+		+	27%
3		Peking University	8	1190	+	+		+	+	+	+	+	+		+	42%
4		The University of Tokyo	8	1566	+	+		+	+	+	+	+	+		+	68%

☆ Star

排名变化趋势



排名

5		Seoul National University	7	719	+	+		-	+	+		+	+		+	41%
6		University of New South Wales	7	855	+	+		+	+	+	+	+	+		+	30%
7		Tsinghua University	7	1002	+	+			+	+		+	+		+	50%
8		Shanghai Jiao Tong University	7	1016	+	+			+	+	-	+	+		+	50%
9		St. Petersburg ITMO University	7	1086	+	+			+	+	+	+	+		+	66%
10		University of Central Florida	7	1133	+	+			+	+	+	+	+		+	56%
11		Massachusetts Institute of Technology	7	1144	+	+			+	+	+	+	+		+	63%
12		Vilnius University	7	1174	+	+			+	+	+	+	+		+	56%
13		Ural Federal University	7	1234	+	+			+	+	+	+	+		+	53%
14		University of Warsaw	6	649	+	+			+	+	+	+	+		+	45%
15		University of Engineering and Technology - VNU	6	757	+	+			+	+	+	+	+		+	50%
16		Taiwan University	6	806	+	+			+	+	+	+	+		+	45%
17		Universidade de São Paulo	6	810	+	+			+	+	+	+	+		+	53%

关于 XCPCIO Board 的更多详细信息, 欢迎访问 <https://board.xcpcio.com>。

现在, XCPCIO 希望开发一个 Board 的 CLI 工具, 希望您能帮助他们。

首先, XCPCIO 会以一定的格式将队伍数据和提交数据提供给你(具体的数据格式请参考 Input)。

其次, 你需要回答用户的 q 个询问, 询问类型如下:

- `teamstatus timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳为 `timestamp` 的时刻的队伍状态, 包括排名、罚时、每道题的 AC 状态、Dirt 率(具体可参考 Output)。
- `minrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最小是多少。
- `maxrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最大是多少。
- `account timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的 AC 数量。

- `submitcount timestamp problemid`, 表示询问编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的提交数量。
- `teamcount timestamp problemnum`, 表示询问在时间戳小于等于 `timestamp` 之前的比赛过程中 AC 了 `problemnum` 道题目的队伍数量。

Dirt 率: 所有 AC 了的题目中的 $\lfloor \frac{\text{非 AC 提交数}}{\text{总提交数}} \rfloor$ 。如果还没有 AC 题目, 那么你需要输出 “N/A”。

关于罚时的计算:

- 只有 AC 的题目才会计入罚时。
- 一道题目 AC 后, 它的计算公式为 AC 时间 + 非 AC 的提交数量 $\cdot 20$ 。

关于排名:

- 首先按 AC 题目数量从大到小排名。
- AC 数量相同的情况下, 其次按罚时从小到大排名。
- AC 数量和罚时均相同的情况下, 最后按队伍编号从小到大排名。

容易证明, 按照以上排名规则, 不会出现两支队伍排名一致的情况。

并且我们约定在比赛的第 0 时刻, 没有任何提交, 每支队伍的排名就是它的队伍编号。

在某一时刻, 如果有多发提交, 那么在这一时刻整个排行榜如果按一发一发的提交来变化可能会变化很多次。

但是我们在此处约定, 如果某一时刻有多发提交, 那么这一时刻的排名是基于这些提交产生的结果都应用之后。

也就是说, 在一场比赛中, 榜单的结果只有 T 种, 当时间戳为 `timestamp` 时的榜单为所有时间戳小于等于 `timestamp` 的提交产生的结果榜单。

Input

单组数据评测。

第一行包含 4 个正整数 $n(1 \leq n \leq 500)$, $T(1 \leq T \leq 10^5)$, $m(1 \leq m \leq 10^6)$, $p(1 \leq p \leq 13)$, 表示该场比赛总的队伍数, 总的时间, 总的提交数和题目数量。

接下来 m 行, 每行表示一个提交。包含 3 个正整数 $team_i(1 \leq team_i \leq n)$, $problemid_i(1 \leq problemid_i \leq p)$, $timestamp_i(1 \leq timestamp_i \leq T)$, 表示队伍编号, 题目编号和时间戳, 以及一个字符串 `AC|WA`, 表示提交的状态(AC 表示正确通过, WA 表示解答错误)。

第 $m+1$ 行包含一个正整数 $q(1 \leq q \leq 10^5)$, 表示询问总数。

接下来 q 行, 每行表示一个询问。

- `teamstatus timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳为 `timestamp` 的时刻的队伍状态, 包括排名、罚时、每道题的 AC 状态、Dirt 率(具体格式可参考 Output)。
- `minrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最小是多少。

- `maxrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最大是多少。
- `account timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的 AC 数量。
- `submitcount timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的提交数量。
- `teamcount timestamp problemnum`, 表示询问在时间戳小于等于 `timestamp` 之前的比赛过程中 AC 了 `problemnum` 道题目的队伍数量。

对于以上所有询问类型, 都有 $1 \leq \text{timestamp} \leq T$, $1 \leq \text{teamid} \leq n$, $1 \leq \text{problemid} \leq p$, $1 \leq \text{problemnum} \leq p$ 。

一支队伍如果在 AC 一道题目后继续对该题进行提交, 我们认为 AC 之后的提交对数据统计分析没有意义, 所以我们提前剔除了这些「脏提交」, 或者你可以认为我们提供的数据中不存在这样的提交。

并且数据保证, 每支队伍在每一时刻, 对于某一道题最多只有一发提交。

需要注意的是, 给出的 m 条提交并不一定是按 `timestamp` 排序的。

Output

对于每一个询问, 输出包含一行。

- `teamstatus`, 首先是一个正整数表示队伍的排名, 接着一个长度为 p 的字符串表示题目的 AC 情况, 用 `[` 和 `]` 包裹, 并且我们约定用 `o` 表示 AC, 用 `x` 表示提交过并且 WA 了, 用 `-` 表示还未提交, 接着有一个正整数和一个百分号表示 Dirt 率, 注意这三部分内容需要用空格间隔, 具体可参考样例。
- `minrank`, 输出包含一行一个整数表示该队伍的最小排名数值。
- `maxrank`, 输出包含一行一个整数表示该队伍的最大排名数值。
- `account`, 输出包含一行一个整数表示该题目的 AC 数量。
- `submitcount`, 输出包含一行一个整数表示该题目的提交数量。
- `teamcount`, 输出包含一行一个整数表示 AC 了 `problemnum` 道题目的队伍数量。

Example

standard input	standard output
5 10 10 13	4 [-----] N/A
1 1 5 WA	2 [o-----] 0%
1 1 6 AC	3 [-----o-----] 0%
2 1 1 AC	5 [-----] N/A
3 5 5 WA	1 [oo-----] 33%
3 5 6 AC	1
3 7 1 AC	5
3 13 10 AC	5
5 1 1 AC	2
5 2 1 WA	2
5 2 2 AC	2
21	2
teamstatus 1 1	0
teamstatus 2 2	1
teamstatus 3 3	3
teamstatus 4 4	2
teamstatus 5 5	1
minrank 1 1	2
maxrank 1 5	1 [----o-o-----o] 25%
maxrank 2 5	1 [o-----] 0%
account 1 1	4 [x-----] N/A
account 2 1	
submitcount 1 1	
submitcount 2 1	
submitcount 9 13	
submitcount 10 13	
teamcount 1 1	
teamcount 2 1	
teamcount 3 2	
teamcount 10 1	
teamstatus 10 3	
teamstatus 1 2	
teamstatus 5 1	

Problem B. BM 算日期

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

BM 是一位数学高手，今天他迎来了 Kita 的挑战。Kita 想让 BM 算出这几年内有多少个闰年。

BM 觉得这问题实在太简单了，于是 Kita 加大了难度。

他先给出第一个年份，再给出一个整数。Kita 要 BM 进行加法运算后得到第二个年份，然后算出这两个年份之间有多少个闰年。

然而，BM 对于大于 9999 的年份十分恐惧，于是他耗尽力气，不仅将年份大于 9999 的部分删除了，并且利用多余的部分使这个年份更小了。

例如，如果 Kita 给出的第一个年份为 9997，整数为 3，进行加法运算后得到的年份为 10000，大于 9999 的部分为 1 年，所以得到的第二个年份为 $9999 - 1 = 9998$ ，BM 最终得到的年份区间就是 [9997, 9998]。

在进行了如上操作后，BM 筋疲力尽，甚至没有力气确定自己最终的年份区间是什么。为了守护他的数学高手的称号，请您帮助他算出最终的答案。

Input

第一行包含一个整数 $T(1 \leq T \leq 10^2)$ ，表示有 T 组输入。

每组输入在一行内给出两个以空格隔开的整数 $Y(1 \leq Y \leq 9999)$, $A(-9998 \leq A \leq 19996)$ ，表示 Kita 给出的年份和整数。

数据保证最终算出来的年份均为正整数。

Output

对于每组输入，输出一行包含一个整数 x ，表示闰年的数量。

Example

standard input	standard output
3	0
9997 3	2424
1 9998	1
9999 -3	

Note

在样例的第 3 组数据中， $9999 - 3 = 9996$ ，所以我们最终确定的区间为 [9996, 9999]，其中 9996 为闰年，所以答案是 1。

闰年是公历中的名词，分为普通闰年和世纪闰年。

普通闰年：公历年份是 4 的倍数的，且不是 100 的倍数，为闰年。（如 2004 年就是闰年）。

世纪闰年：公历年份是整百数的，必须是 400 的倍数才是闰年（如 1900 年不是世纪闰年，2000 年是世纪闰年）。

——源自百度百科

Problem C. BBpigeon Counting Trees

Input file: standard input
 Output file: standard output
 Time limit: 1 second
 Memory limit: 256 megabytes

You're given a tree with n nodes, labeled from 1 to n , node 1 is the root of the given tree.
 The depth of i -th node is the number of nodes in the path between root and the i -th node including itself.
 Given the depth of each node, please calculate the number of different trees meet the description.
 Two trees are considered different iff there exist a node x have different number of sons in these trees.

Input

First line contains a positive integer $n(1 \leq n \leq 10^5)$, the number of nodes.
 Second line contains n integers $a_i(1 \leq n \leq 10^5)$, the i -th integer is the depth of node i .

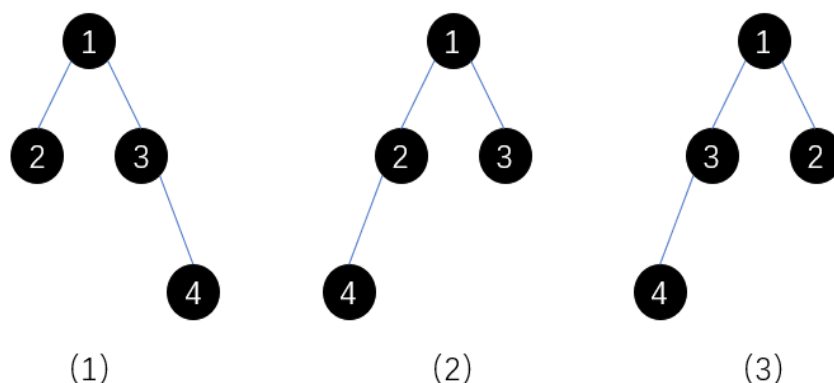
Output

Print the number of different tree staisfing the description above.
 Since the answer could be quite large, you should just module the answer with 998244353.

Examples

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

Note



tree(1) and tree(2) are considered different since 3-th node have different number of sons in these trees.
 tree(1) and tree(3) are the same because all nodes have same number of sons in both trees.

Problem D. Palindrome Hard Problem

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

最近 wifePie 学长在研究字符串难题，对回文问题颇有研究，经常一言不合就掏出回文树和马拉车，并且基于深度研究的基础上提出以下问题：

给出 n 个长度相同的，只包含小写字母的字符串，标号为 1 到 n 。并且我们约定第 i 个字符串为 s_i 。你可以进行不限次数的如下操作，每个操作都会使得字符串个数减 1：

- 把 s_i 拼接到 s_j 的后面，操作后 s_i 消失。
- 把 s_i 拼接到 s_j 的前面，操作后 s_i 消失。
- 将 s_i 和 s_j 进行交错排列，形成一个新的字符串 S 。操作后 S 替代 s_j 的位置，然后 s_i 和 s_j 消失。（例如 “aaaa” 与 “bbbb” 重构后得到的串为 “abababab”）。
- 直接舍弃 s_i 。

经过若干次上述操作后，你可以获得一个终极串，问这个终极串最多能分割成多少个回文串。

值得注意的是，上述操作是由你选择如何操作的，你要保证你最终得到的终极串能够分割出的回文串的数量是最多的。

Input

单组数据评测。

第一行包含一个正整数 $n(1 \leq n \leq 10^5)$ 。

接下来的 n 行，每行有一个长度为 m 的字符串，其中只包含小写字母。

数据保证 $n \cdot m \leq 10^6$ 。

Output

输出包含一个正整数，表示终极串中最多能分割出的回文串的数量。

Examples

standard input	standard output
1 a	1
2 a b	2

Problem E. BM 充饥

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

到了年末，BM 的积蓄所剩无几，但他又很想吃 KFC，请您帮助他画“KFC”充饥吧！

Input

“kfc”.

Output

“kfc” 的字符画。

Example

standard input	standard output
kfc	<pre>-- ----- ___/ ______ / \ ___\ ___\ < \ ___ __ _ \ __ ___ > \ / \ /</pre>

Problem F. kita 买礼物

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

圣诞节快到了, kita 想给他的女神买个礼物。现在 kita 手里一共拥有 n 种不同面额的硬币, a_i 代表每种硬币的面额, b_i 代表第 i 种硬币所拥有的数量。

但是 kita 是个很粗心的人, 他只知道礼物的价格不高于 m 。

现在他想知道他手中的硬币能够组合成 $1 \sim m$ 中的几种情况。

Input

第一行输入 $n(1 \leq n \leq 100)$, $m(1 \leq m \leq 10^5)$, 分别代表硬币种类数和礼物的价格。

第二行输入 n 个数字, 第 i 个数字 $a_i(0 \leq a_i \leq 10^5)$ 代表第 i 种硬币的面额。

第三行输入 n 个数字, 第 i 个数字 $b_i(0 \leq b_i \leq 10^5)$ 代表第 i 种硬币所拥有的数量。

Output

输出 $1 \sim m$ 中所能够组合的情况的个数。

Example

standard input	standard output
3 10 1 2 4 2 1 1	8

Problem G. New Game

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

新的游戏开始了，所有人都想取得胜利以获得更高的排名。

但由于条件受限不能给任意两个人都安排对决，因此每场比赛的输赢直接关乎最后的排名。

已知每场比赛两名选手的输赢结果，求游戏最后的排名。

游戏的排名规则为如果 A 赢 B, B 赢 C 则最后排名为 A, B, C。

也就是说，如果 A 赢了 B，那么 A 的最终排名一定排在 B 的前面。

Input

第一行一个正整数 $T(1 \leq T \leq 10)$ ，表示有 T 组数据。

每组数据中的第一行包含两个整数 $n(1 \leq n \leq 10^3)$, $m(0 \leq m \leq n^2)$ ，其中 n 表示参加游戏的总人数， m 表示有 m 条比赛结果。

接下来的 m 行数据中，每行也有两个整数 $A, B(1 \leq A, B \leq n)$ 表示这次对局中玩家 A 赢了玩家 B。

数据中可能存在两条一样的比赛结果。

Output

给出一个符合要求的排名。输出时玩家之间有空格，最后一名后面没有空格。

其他说明：符合条件的排名可能不是唯一的，此时要求输出时编号小的玩家在前。

输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名，即不存在 A 赢了 B, B 赢了 C, C 赢了 A 的情况。

Example

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

Problem H. Hsueh- Draw Progress

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

According to scientific research, in the waiting process, if the progress bar can be used to show the progress, the waiting person will be more patient.

Therefore, Hsueh- needs you to help him draw a progress bar in command line.

The specific progress bar style can be obtained by observing the sample.

Input

There are several test cases.

The first line contains a single integer $T(1 \leq T \leq 10)$, denoting the number of test cases. Then follow all the test cases.

For each test case, the first line contains two integers $n(1 \leq n \leq 10^5)$ and $m(0 \leq m \leq n)$, denoting the total number of processes and number of completed processes.

Output

For each test case, you first need to draw a progress bar and then you need to output the percentage of progress. In other words, the percentage of progress is $\lfloor \frac{m}{n} \rfloor$.

It is worth noting that, in the progress bar, we use “#” to indicate completed processes, and “-” to indicate unfinished processes.

Example

standard input	standard output
6	[##-----] 20%
10 2	[#####-----] 33%
15 5	[###-----] 33%
9 3	[##--] 50%
4 2	[-----] 0%
5 0	[#####] 100%
10 10	

Problem I. BM 旅游

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

众所周知 bm 学姐喜欢旅游。这一天他来到了著名的杭师大景点勤园 13-406。该景点中有四个著名的观景台分为 A, B, C, D。

- 如果只去过上述一个观景台, 输出 “Oh dear!!”(没有引号)。
- 如果去过两个, 输出 “BaoBao is good!!”(没有引号)。
- 如果去过三个, 输出 “Bao Bao is a SupEr man:///!”(没有引号)。
- 如果全都去过, 输出 “Oh my God!!!!!!!!!!!!!!!!!!!!!!”(没有引号)。
- 如果一个都没去过, 输出 “Bao Bao is so Zhai.....”(没有引号)。

Input

第一行包含 4 个整数 A, B, C, D ($0 \leq A, B, C, D \leq 10^{18}$)。

A 的每个位的数字和若大于等于 16 或者等于 6 则说明去过观景台 A 。

B, C, D 亦如此。

Output

根据题干提示输出 5 种字符串之一。

Example

standard input	standard output
1 2 3 4	Bao Bao is so Zhai.....

Note

黄鸡学长老寒手, 打字的时候老是手抖, 请注意输出字符串的格式。

Problem J. 大扫除

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

你是一个住在大豪宅里的富二代「聘请的老青结清洁工」。

你需要给这座高 n 层楼的豪宅清理垃圾。

已知字符 “.” 表示没有垃圾，其他任何字符都表示垃圾，每种字符表示一种垃圾。

要求计算整个豪宅每层楼垃圾种类的总和。

Input

第一行包含一个整数 T ($1 \leq T \leq 10$)，表示有 T 组数据。

每组数据的第一行有一个数字 n ($1 \leq n \leq 10^2$)，表示豪宅的层数。

接下来 n 行每行有一个字符串 s ($1 \leq |s| \leq 10^6$) 这层楼的垃圾情况。

数据保证字符串中没有不可见字符，也没有空字符串。

数据保证总的字符数不超过 10^7 。

Output

每组数据输出一个整数表示整座豪宅每层楼的垃圾总类的和。

Example

standard input	standard output
2	2
2	2
...#	
#...	
3	
....	
@#@#@	
.....	

Note

对于第一组样例，“#” 分别在两层中，所以被统计两次。

需要注意的是，统计的是每层楼垃圾种类的总和，而不是整栋楼的垃圾种类。

Problem K. Kwords Find Kth Element

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Hello, guys. I am **Kwords**. I prepared a problem for the contest invited by Hsueh-.

Due to I was very boring, I generate n arrays, labeled from 1 to n . And then I will give q queries.

In each queries:

- First, I will give a interger m , and then m different intergers.
- You need to merge the arrays pointed to by these m labels which I given.
- Then, I will give a interger k .
- You need to answer me, what is the k -th smallest number in the combined array.

It is attention that, each query is independent. It means that the array merging operation of a single query will not affect other queries.

Input

The first line contains a single interger $n(1 \leq n \leq 10^5)$, denoting the number of arrays.

The next n lines, for each line:

- First give a single integer $m_i(1 \leq m_i \leq 10^5)$, denoting the size of the i -th array.
- Followed by m_i integers $a_{i,j}(1 \leq a_{i,j} \leq 10^9)$, denoting the j -th number in the i -th array.

The $n + 1$ line contains a single integer $q(1 \leq q \leq 10^5)$, denoting the number of queriyes.

The next q liens, for each line:

- First give a single integer $l_i(1 \leq l_i \leq 10^5)$, denoting the size of the i -th index query.
- Followed by l_i distinct integers $b_{i,j}(1 \leq b_{i,j} \leq n)$, denoting the j -th index in the i -th query.
- Last will given a single interge $k_i(1 \leq k_i \leq \sum_{j=1}^{l_i} array[b_j].size())$, denoting you need to find k -th smallest element in the combined array.

It is guaranteed that $\sum_{i=1}^n m_i \leq 10^5$ and $\sum_{i=1}^q l_i \leq 10^5$.

Output

For each query, you should print the k -th smallest element in the combined array in a single line.

Example

standard input	standard output
5	3
1 1	5
2 2 3	58
3 5 10 6	1
4 4 58 2 1	4
5 1000000000 9 8 4 5	
5	
1 2 2	
2 2 3 3	
3 3 4 5 11	
4 5 4 3 2 1	
5 1 2 5 4 3 7	

Problem L. 咆哮

Input file: standard input
 Output file: standard output
 Time limit: 1 second
 Memory limit: 256 megabytes

lanChe 最近迷上了三国杀国战。其中有一局，使用了如下两个英雄大杀四方。



lanChe 在其中一回合中杀穿了 4 个人，并且想把这份喜悦分享给你们。但这一局的情况说不清楚，便换个思路把喜悦分享给你们。

现在 lanChe 处于鏖战当中。根据国战的规则，鏖战时候不能使用桃进行回血，改成把桃当成杀或者闪。

lanChe 上回合惹怒了对面两个阵营，现在吴国和群国联合起来对抗蜀国。对面 2 名角色的阵容如下：





这场鏖战如宁所料，将会是场美丽的世界名画。为了赢取蜀国的最终胜利，lanChe 现在想知道，他将要承受多少伤害。现在假设一下，假设在第 x 轮，lanChe 在这个回合内最终能拿到 x 张杀，为简化起见，现规定每轮的杀全部出完，并且只能选一名角色。

给出吴国阵营的血量 HP_w 和他每轮对蜀国造成的固定伤害值 ATK_w ，以及群国阵营的血量 HP_q 和他每轮对蜀国造成的固定伤害值 ATK_q 。

一旦某国阵营血量 ≤ 0 ，直接出局，下轮不再对蜀国造成伤害。

现在 lanChe 想知道赢取蜀国的胜利最少需要承受多少伤害。

并且制定一份攻击策略使得自身承受的伤害最小。策略的格式为：

- 第 i 轮若攻击吴国阵营，则在第 i 位输出 ‘W’。
- 第 i 轮若攻击群国阵营，则在第 i 位输出 ‘Q’。

例如，在第 1 轮攻击吴国，第二轮攻击群国，则输出字符串 ‘WQ’。

lanChe 现今在苦苦防守当中，没心思去思考这份完美的策略，便请求你的帮助。

答案输出最终承受的最小伤害以及相应的策略。若有多份策略，请输出字典序最小的策略。

Input

第一行输入 $T (1 \leq T \leq 10^4)$ ，代表有 T 组数据。

接下来 T 行，每行有四个正整数 HP_q, HP_w, ATK_q, ATK_w ，表示群国当前的血量，吴国当前的血量，群国对蜀国造成的伤害以及吴国对蜀国造成的伤害。 ($1 \leq HP_q, HP_w, ATK_q, ATK_w \leq 10^9$)。

Output

对于每组数据，输出一行，表示蜀国赢取胜利需要承受的最小伤害值以及蜀国的攻击策略，两者之间以空格间隔。

若有多种策略选择，则输出字典序最小的策略。

Example

standard input	standard output
2	45 WWQ
3 3 5 15	203 QWQ
4 2 1 100	

Note

字典序的定义：按照第一个字母, 以 **a**、**b**、**c**...**z** 的顺序排列。如果第一个字母一样, 那么比较第二个、第三个乃至后面的字母。如果比到最后两个单词不一样长 (比如 **sigh** 和 **sight**), 那么把短者排在前。

——源自百度百科

Problem M. XCPCIO Board CLI Hard

Input file: standard input
 Output file: standard output
 Time limit: 5 seconds
 Memory limit: 256 megabytes

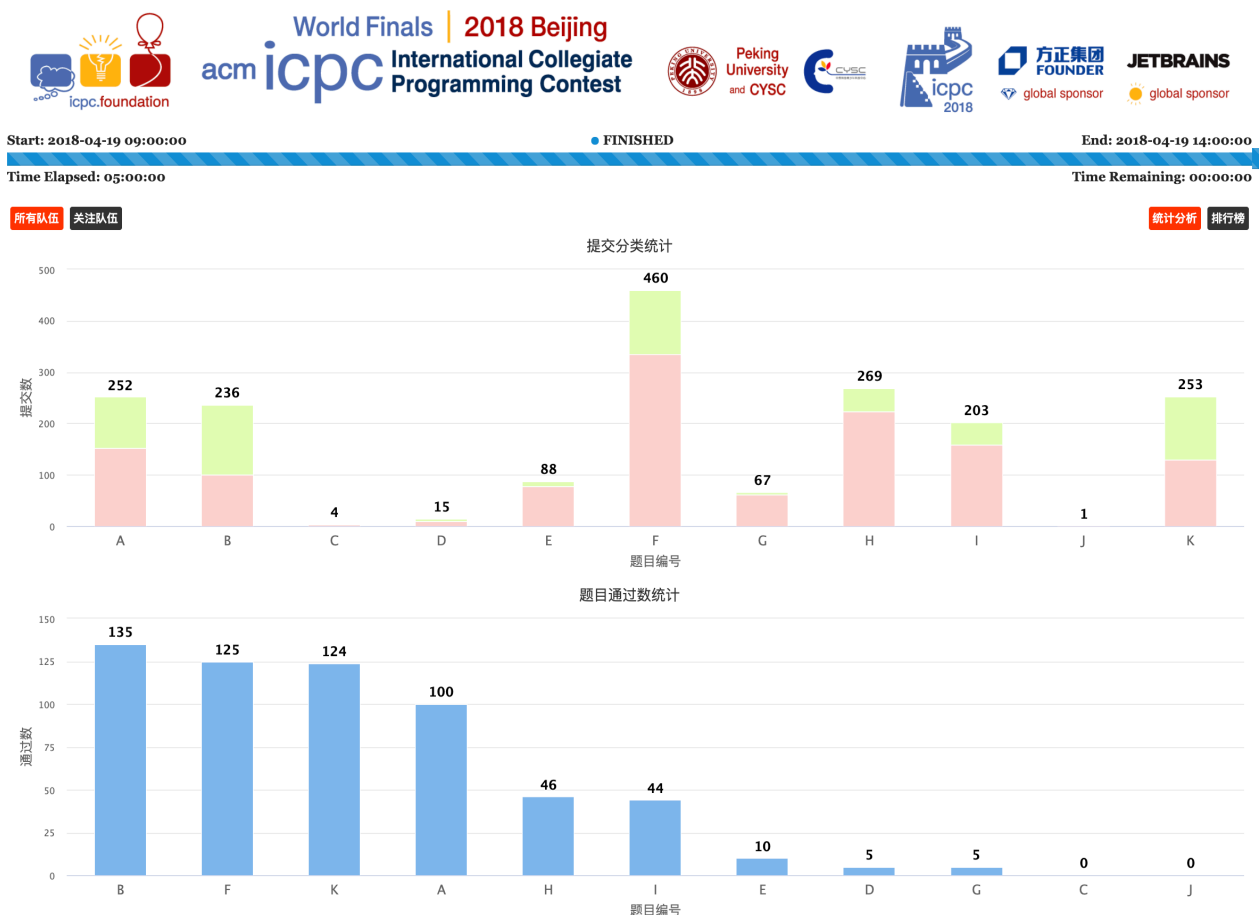
本题和 *easy version* 的区别仅仅在于队伍总数 n 的范围不同，其它均一致。

没有榜单的 *CPC 比赛怎么能称之为比赛？

Hsueh-

众所周知，在 *CPC 比赛中，榜单是一个很重要的部分。

最近，在 ICPC 和 CCPC 比赛的举办期间，有一个公益组织在 GitHub 上创建了一个项目，用于渲染 *CPC 的榜单，致力于提供更丰富的关于榜单的数据统计分析。



除了数据统计分析，XCPCIO Board 还提供了如下特性：

- 通过拖动进度条来复现某一时刻的榜单状态。
- 点开每支队伍能够显示该队伍的排名关于时间的变化情况。
- ...

Start: 2018-04-19 09:00:00

FINISHED

End: 2018-04-19 14:00:00

Time Elapsed: 05:00:00

Gold Silver Bronze Honorable First to solve problem Solved problem Attempted problem Pending judgement

Time Remaining: 00:00:00

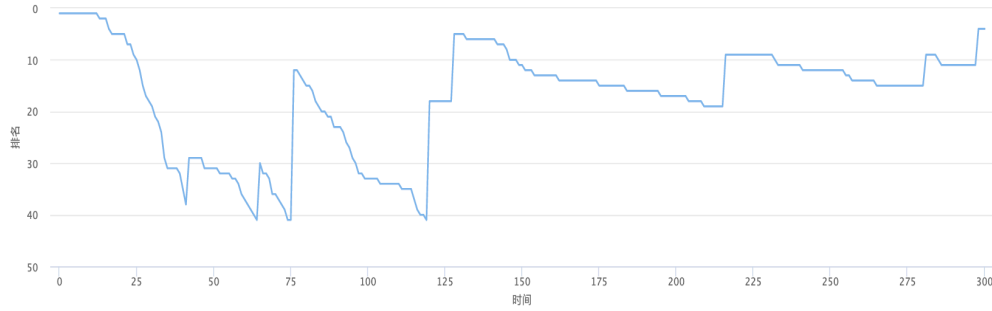
所有队伍 关注队伍

统计分析 排行榜

Place	Badge	Team	Solved	Time	A 100	B 135	C 0	D 5	E 10	F 125	G 5	H 46	I 44	J 0	K 124	Dirt
1		Moscow State University	9	1427	+	+		+	+	+	+	+	+		+	52%
2		Moscow Institute of Physics & Technology	8	960	+	+		+	+	+	+	+	+		+	27%
3		Peking University	8	1190	+	+		+	+	+	+	+	+		+	42%
4		The University of Tokyo	8	1566	+	+		+	+	+	+	+	+		+	68%

☆ Star

排名变化趋势



排名

5		Seoul National University	7	719	+	+		-	+	+		+	+		+	41%
6		University of New South Wales	7	855	+	+		+	+	+	+	+	+		+	30%
7		Tsinghua University	7	1002	+	+			+	+		+	+		+	50%
8		Shanghai Jiao Tong University	7	1016	+	+			+	+		+	+		+	50%
9		St. Petersburg ITMO University	7	1086	+	+			+	+		+	+		+	66%
10		University of Central Florida	7	1133	+	+			+	+		+	+		+	56%
11		Massachusetts Institute of Technology	7	1144	+	+			+	+		+	+		+	63%
12		Vilnius University	7	1174	+	+			+	+		+	+		+	56%
13		Ural Federal University	7	1234	+	+			+	+		+	+		+	53%
14		University of Warsaw	6	649	+	+			+	+		+	+		+	45%
15		University of Engineering and Technology - VNU	6	757	+	+			+	+		+	+		+	50%
16		Taiwan University	6	806	+	+			+	+		+	+		+	45%
17		Universidade de São Paulo	6	810	+	+			+	+		+	+		+	53%

关于 XCPCIO Board 的更多详细信息, 欢迎访问 <https://board.xcpcio.com>。

现在, XCPCIO 希望开发一个 Board 的 CLI 工具, 希望您能帮助他们。

首先, XCPCIO 会以一定的格式将队伍数据和提交数据提供给你(具体的数据格式请参考 Input)。

其次, 你需要回答用户的 q 个询问, 询问类型如下:

- `teamstatus timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳为 `timestamp` 的时刻的队伍状态, 包括排名、罚时、每道题的 AC 状态、Dirt 率(具体可参考 Output)。
- `minrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最小是多少。
- `maxrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最大是多少。
- `account timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的 AC 数量。

- `submitcount timestamp problemid`, 表示询问编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的提交数量。
- `teamcount timestamp problemnum`, 表示询问在时间戳小于等于 `timestamp` 之前的比赛过程中 AC 了 `problemnum` 道题目的队伍数量。

Dirt 率: 所有 AC 了的题目中的 $\lfloor \frac{\text{非 AC 提交数}}{\text{总提交数}} \rfloor$ 。如果还没有 AC 题目, 那么你需要输出 “N/A”。

关于罚时的计算:

- 只有 AC 的题目才会计入罚时。
- 一道题目 AC 后, 它的计算公式为 AC 时间 + 非 AC 的提交数量 $\cdot 20$ 。

关于排名:

- 首先按 AC 题目数量从大到小排名。
- AC 数量相同的情况下, 其次按罚时从小到大排名。
- AC 数量和罚时均相同的情况下, 最后按队伍编号从小到大排名。

容易证明, 按照以上排名规则, 不会出现两支队伍排名一致的情况。

并且我们约定在比赛的第 0 时刻, 没有任何提交, 每支队伍的排名就是它的队伍编号。

在某一时刻, 如果有多发提交, 那么在这一时刻整个排行榜如果按一发一发的提交来变化可能会变化很多次。

但是我们在此处约定, 如果某一时刻有多发提交, 那么这一时刻的排名是基于这些提交产生的结果都应用之后。

也就是说, 在一场比赛中, 榜单的结果只有 T 种, 当时间戳为 `timestamp` 时的榜单为所有时间戳小于等于 `timestamp` 的提交产生的结果榜单。

Input

单组数据评测。

第一行包含 4 个正整数 $n(1 \leq n \leq 10^5)$, $T(1 \leq T \leq 10^5)$, $m(1 \leq m \leq 10^6)$, $p(1 \leq p \leq 13)$, 表示该场比赛总的队伍数, 总的时间, 总的提交数和题目数量。

接下来 m 行, 每行表示一个提交。包含 3 个正整数 $team_i(1 \leq team_i \leq n)$, $problemid_i(1 \leq problemid_i \leq p)$, $timestamp_i(1 \leq timestamp_i \leq T)$, 表示队伍编号, 题目编号和时间戳, 以及一个字符串 `AC|WA`, 表示提交的状态(AC 表示正确通过, WA 表示解答错误)。

第 $m+1$ 行包含一个正整数 $q(1 \leq q \leq 10^5)$, 表示询问总数。

接下来 q 行, 每行表示一个询问。

- `teamstatus timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳为 `timestamp` 的时刻的队伍状态, 包括排名、罚时、每道题的 AC 状态、Dirt 率(具体格式可参考 Output)。
- `minrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最小是多少。

- `maxrank timestamp teamid`, 表示询问队伍编号为 `teamid` 的队伍在时间戳小于等于 `timestamp` 之前的时间的比赛过程中排名数值最大是多少。
- `account timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的 AC 数量。
- `submitcount timestamp problemid`, 表示询问题目编号为 `problemid` 的题目在时间戳小于等于 `timestamp` 之前的比赛过程中总的提交数量。
- `teamcount timestamp problemnum`, 表示询问在时间戳小于等于 `timestamp` 之前的比赛过程中 AC 了 `problemnum` 道题目的队伍数量。

对于以上所有询问类型, 都有 $1 \leq \text{timestamp} \leq T$, $1 \leq \text{teamid} \leq n$, $1 \leq \text{problemid} \leq p$, $1 \leq \text{problemnum} \leq p$ 。

一支队伍如果在 AC 一道题目后继续对该题进行提交, 我们认为 AC 之后的提交对数据统计分析没有意义, 所以我们提前剔除了这些「脏提交」, 或者你可以认为我们提供的数据中不存在这样的提交。

并且数据保证, 每支队伍在每一时刻, 对于某一道题最多只有一发提交。

需要注意的是, 给出的 m 条提交并不一定是按 `timestamp` 排序的。

Output

对于每一个询问, 输出包含一行。

- `teamstatus`, 首先是一个正整数表示队伍的排名, 接着一个长度为 p 的字符串表示题目的 AC 情况, 用 `[` 和 `]` 包裹, 并且我们约定用 `o` 表示 AC, 用 `x` 表示提交过并且 WA 了, 用 `-` 表示还未提交, 接着有一个正整数和一个百分号表示 Dirt 率, 注意这三部分内容需要用空格间隔, 具体可参考样例。
- `minrank`, 输出包含一行一个整数表示该队伍的最小排名数值。
- `maxrank`, 输出包含一行一个整数表示该队伍的最大排名数值。
- `account`, 输出包含一行一个整数表示该题目的 AC 数量。
- `submitcount`, 输出包含一行一个整数表示该题目的提交数量。
- `teamcount`, 输出包含一行一个整数表示 AC 了 `problemnum` 道题目的队伍数量。

Example

standard input	standard output
5 10 10 13	4 [-----] N/A
1 1 5 WA	2 [o-----] 0%
1 1 6 AC	3 [-----o-----] 0%
2 1 1 AC	5 [-----] N/A
3 5 5 WA	1 [oo-----] 33%
3 5 6 AC	1
3 7 1 AC	5
3 13 10 AC	5
5 1 1 AC	2
5 2 1 WA	2
5 2 2 AC	2
21	2
teamstatus 1 1	0
teamstatus 2 2	1
teamstatus 3 3	3
teamstatus 4 4	2
teamstatus 5 5	1
minrank 1 1	2
maxrank 1 5	1 [----o-o-----o] 25%
maxrank 2 5	1 [o-----] 0%
account 1 1	4 [x-----] N/A
account 2 1	
submitcount 1 1	
submitcount 2 1	
submitcount 9 13	
submitcount 10 13	
teamcount 1 1	
teamcount 2 1	
teamcount 3 2	
teamcount 10 1	
teamstatus 10 3	
teamstatus 1 2	
teamstatus 5 1	