Problem A. 旷野之息

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Cuber QQ 终于打败盖农救回了塞尔达公主，海拉鲁大地也开始灾后重建。

在统计学中，幂律表示的是两个量之间的函数关系，其中一个量的相对变化会导致另一个量的相应幂次比例的变化，且与初值无关：表现为一个量是另一个量的幂次方。例如，正方形面积与边长的关系，如果长度扩大到两倍，那么面积扩大到四倍。

幂律的涉及范围极其广泛，各种各样的物理、生物和人为现象的分布在大致遵循着幂律，包括月球表面月坑的大小，太阳耀斑的强度，各物种的觅食模式，神经元集群活动模式的规模，大多数语言的用词频率等等。

帕累托法则是幂律分布表现出的重要特征，其指出，约仅有 20% 的因素影响 80% 的结果。也就是说：所有变因中，最重要的仅有 20% ，虽然剩余的 80% 占了多数，影响的幅度却远低于 “关键的少数”。例如，“全球 80 % 的财富被 20 % 的人所占据”。

现在 Cuber QQ 也想在海拉鲁验证帕累托法则，他调查了海拉鲁大地上居民们拥有的财富数量，你需要找到两个数 x 和 y，使得调查中的数据满足 $x\%$ 的人拥有了 $y\%$ 的财富，并且要求 $y-x$ 的尽可能大，以方便 Cuber QQ 验证帕累托法则的结论。

Input

输入第一行包含一个整数 $n(1 \le n \le 10^5)$ ，表示被调查居民的数量。

第二行包含 n 个整数 $a_1, a_2, \cdots, a_n (1 \le a_i \le 10^5)$ 表示每个人拥有的财富总额。

Output

输出一行一个浮点数，表示答案，即最大的 $y-x$。输出的答案和标准结果的相对误差和绝对误差在 10^{-6} 之内会被认为是正确的。

Example

standard input	standard output
5 1 3 5 4 2	20.000000000

Problem B. 希卡之石

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

希卡之石包含了日志记录的功能，主要通过日志文件的形式存储。

日志文件是用于记录系统操作事件的记录文件或文件集合，可分为事件日志和消息日志。具有处理历史数据、诊断问题的追踪以及理解系统的活动等重要作用。日志文件应用广泛，在 linux、Windows 操作系统以及各大应用程序中均有涉及。本题正是参考 python 语言的 logging 日志的中诸多格式而展开的。具体来说，在 python 中 logging 会生成一个多行的日志文件；其中的每行均为一条日志。每条日志往往又由以下几个元素组成：

%(name)s：logger 的名字；

%(levelno)s：日志级别数值；

%(levelname)s：日志级别名称；

%(pathname)s：当前执行程序的路径；

%(filename)s：当前执行程序名；

%(lineno)d：日志的当前行号；

%(asctime)s：日志的时间；

%(thread)d：线程 ID；

%(threadName)s：线程名称；

%(process)s：进程 ID；

%(message)s：日志信息；

需要注意以下几点：

首先，上述元素各自在一条日志中最多仅出现一次；其次，上述元素中的某一项或者某几项可能没有在一条日志中出现；最后，除了`%(message)s`——也即日志信息——一定会出现在一条日志结尾以外，其他所有元素的顺序是不定的。

另外，除了上述元素之外，一条日志中也可能出现上述元素外部的空格，日志可能会用一个或多个空格把上述元素分隔开。特别地，由于已经使用文中最末冒号作为日志信息这一元素的分隔与标识，所以并不在这一元素之前设置额外的空格来进行分隔。

下面给出适用于本题的、每个日志元素的格式定义以及出现情况定义：

`%(name)s`:

`logger`，也即日志器的名字，其基本格式为：$logger : [a-z\_A-Z]+$（不含空格）；

例如：$logger : root$。

`%(levelno)s`:

日志级别数值与下文所述日志级别名称一一对应，仅仅会出现五种数字，分别是 10,20,30,40,50。它们分别与 $DEBUG$，$INFO$，$WARNING$，$ERROR$，$CRITICAL$ 对应。

`%(levelname)s`:

$python.logging$ 日志文件的日志级别名称元素仅仅会出现以下五种字样：

$DEBUG$：用来打印调试信息，级别最低；

$INFO$：用来打印一些正常操作信息；

$WARNING$：用来打印警告；

$ERROR$：用来打印错误；

$CRITICAL$：用来打印一些致命错误信息，等级最高。

`%(pathname)s`:

完整的路径名称的基本格式为：$[A-Z] : (/[A-Za-z\_0-9][A-Za-z\_0-9]*) + .[a-z]+$(注意正则表达式中的空格：文件名与文件夹名允许空格, 但保证不会以空格开头)；

例如：$C : /Program\ Files/AMD/atikmdag\_dce.log$。

`%(filename)s`:

执行程序名的基本格式为：$[A-Za-z\_0-9][A-Za-z\_0-9] * .[a-z]+$(注意正则表达式中的空格：文件名允许空格, 但保证不会以空格开头)；

例如：$atikmdag\_dce.log$；

当一条日志同时出现路径名和执行程序名的时候，保证执行程序名恰是路径名的末段（如 C:/pyth.py 与 pyth.py 的关系）

需要注意的是，题目保证：在数据中，符合日志级别名称与日志级别数值格式的字串不会成为执行程序名的前缀。

`%(lineno)d`:

日志的当前行号的基本格式为：$line : [1-9][0-9]*$；

简单地说，这一项由固定字样 $line$: 与其后的一个非零且无前导零整数组成，例如 $line$: 111。

`%(asctime)s`:

日志的时间的基本格式为: $[0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}\ [0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\},[0-9]\{3\}$(注意正则表达式中的空格, 在日期和时间之间);

例如：$1926-08-17\ 01:01:01,111$，其中逗号后面的整数表示毫秒。当然这样粗略地定义可能会产生一些不合理的时间数据，题目保证一年有 12 个月，一天有 24 小时，一小时有 60 分钟，一分钟有 60 秒。当然选手请无视时间数据的不合理之处。

`%(thread)d`:

线程 ID 的基本格式为：$thread : [0-9]+$;

简单地说，这一项由固定字样 $thread$: 与其后的一个可能有前导零的整数组成，例如 $thread$: 01。

`%(threadName)s`:

线程名称的基本格式为：$thread : [a-z\_A-Z]+[0-9]+$，也即由前段固定的前缀，中段字母与后段数码两个部分组成的字串;

例如：$thread : MaIn0123.$

`%(process)s`:

进程 ID 的基本格式为：$process : [0-9]+$;

简单地说，这一项由固定字样 $process$: 与其后的一个可能有前导零的整数组成，例如 $process$: 01。

`%(message)s`:

日志信息这一项必然会出现在每一条日志中，且必然会出现在每条日志的末尾。此外日志信息这一项必定以一个 ":" 开头，且其内部不包含 ":"。换言之一条日志中最后一个 ":" 后面跟随的全部内容必定为日志信息项。

现在 Cuber QQ 希望你能读取多条日志，对每条日志分析其内容，输出其有哪些元素以及各元素的具体内容。

Input

输入多条日志，每条日志占一行，以回车隔开。数据保证输入的日志格式内容合乎题目描述。

Output

对每条日志输出多行:

第一行输出日志编号，从 1 号开始;

从第二行开始，按日志中出现顺序，每行输出一个此条日志中出现的元素;

输出格式形如 "$< type, content >$";

也即用一个尖括号输出一个二元组，第一个项是此日志元素的类型，形如`%(name)s`；第二项是此元素出现在此日志中的具体内容。

Examples

standard input	standard output
logger:root 10 DEBUG C:/program File/a .log a.log line:1 1926-08-17 01:01:01,999 thread:1 thread:main01 process:11:maef__aaoa9208834,.;'p[qaosdif2930 23idfjariwe	1 <%(name)s,logger:root> <%(levelno)s,10> <%(levelname)s,DEBUG> <%(pathname)s,C:/program File/a.log> <%(filename)s,a.log> <%(lineno)d,line:1> <%(asctime)s,1926-08-17 01:01:01,999> <%(thread)d,thread:1> <%(threadName)s,thread:main01> <%(process)s,process:11> <%(message)s,maef__aaoa9208834,.;'p[qaosdif2930 23idfjariwe>
logger:root 10 DEBUG C:/program File/a DEBUG a/logger .log logger .log line: 10 2023-05-17 23:51:00,999 thread:10 thread:DEBUG01 process:11:logger root 10 DEBUG C /program File/a DEBUG a/logger .log logger .log line 10 2023-05-17 23 51 00,999 thread 10 thread DEBUG01 process 11	1 <%(name)s,logger:root> <%(levelno)s,10> <%(levelname)s,DEBUG> <%(pathname)s,C:/program File/a DEBUG a/logger .log> <%(filename)s,logger .log> <%(lineno)d,line:10> <%(asctime)s,2023-05-17 23:51:00,999> <%(thread)d,thread:10> <%(threadName)s,thread:DEBUG01> <%(process)s,process:11> <%(message)s,logger root 10 DEBUG C /program File/a DEBUG a/logger .log logger .log line 10 2023-05-17 23 51 00,999 thread 10 thread DEBUG01 process 11>

Note

数据保证每个测试用例的输入文件的大小均不超过 900 KB。

Problem C. 神庙挑战

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Cuber QQ 在海拉鲁被一个神庙挑战难住了。

八皇后问题是一个以国际象棋为背景的经典问题，如何能够在 8×8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后。

神庙中的难题要比这个问题更难：它将这个问题推广到 n 皇后问题，同时每个皇后吃掉其他皇后规则也进行了修改，每个皇后可以从自己所在的位置的四个斜线方向（左上、右上、左下、右下）中任意选择其中的 d 个方向吃掉其他皇后。

现在神庙挑战中会给出给出 n 和 d，想让 Cuber QQ 尽可能多的在 $n \times n$ 的棋盘中放置皇后，当然 Cuber QQ 可以通过究级手任意指定每一个皇后攻击的 d 个方向，需要保证棋盘中任意的皇后之间不会相互攻击。

Input

输入包含一行两个整数 $n, d(100 \leq n \leq 10^5, 1 \leq d \leq 4)$，表示棋盘大小和每一个皇后可以攻击的方向。

Output

输出第一行包含一个整数 ans，表示最多可以放置的皇后数量。

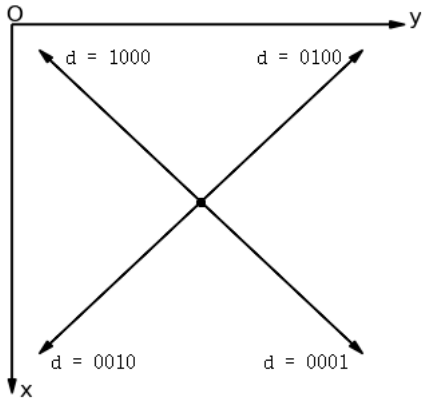
接下来的 ans 行，每行包含三个整数 x, y, z，其中 (x, y) 表示放置皇后的位置，棋盘从左到右从上到下依次编号为 1 到 n，即棋盘左上角为 $(1, 1)$，右下角为 (n, n)。z 是一个不超过 15 的非负整数，可转换成 4 位二进制数 $d = d_3d_2d_1d_0$，d_3, d_2, d_1, d_0 分别表示左上、右上、左下和右下的攻击方向，$d_i = 1$ 表示有效的攻击方向，输出需要保证对于每一个皇后有效的攻击方向恰好为 d 个，且棋盘中任意的皇后之间不会相互攻击。

Example

standard input	standard output
5 2	16 1 1 12 1 5 5 2 1 10 5 2 3 1 2 12 2 5 5 3 1 10 5 3 3 1 3 12 3 5 5 4 1 10 5 4 3 1 4 12 4 5 5 5 1 10 5 5 3

Note

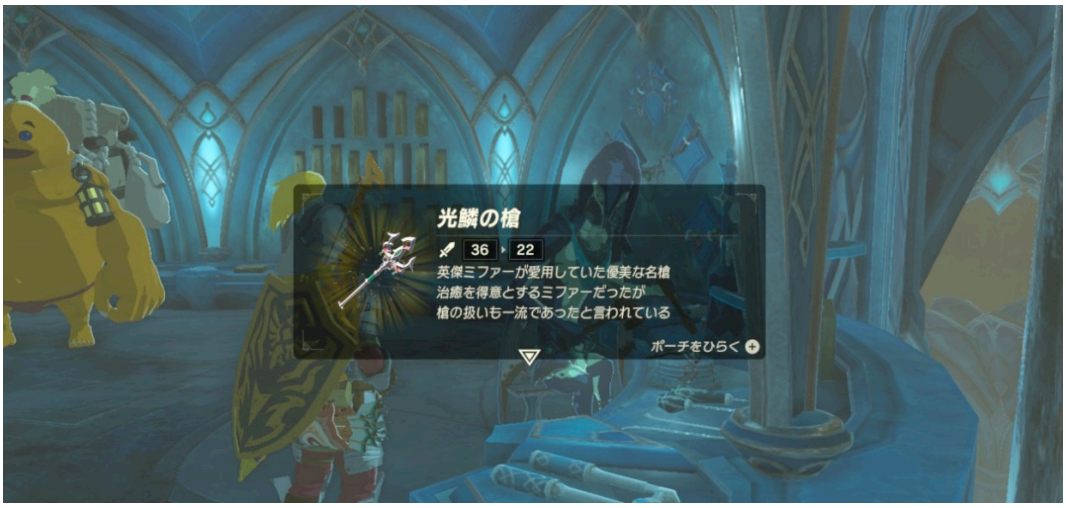
请注意，样例并不是一个合法的在数据范围内的数据，仅作为输入输出样例格式的参考。



Problem D. 光鳞之枪

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

米法留下的武器光鳞之枪有一个非常高仿的复制品“祭祀之枪”，Cuber QQ 拿到两个武器之后总是很难区分它们的区别。经过研究发现，光鳞之枪和祭祀之枪都可以通过编码变成两个字符串，且它们是可以用编辑距离的大小来区分的。



编辑距离，由俄罗斯科学家 Vladimir Levenshtein 在 1965 年提出，也因此而得名 Levenshtein Distance。

在信息论、语言学和计算机科学领域，Levenshtein Distance 是用来度量两个序列相似程度的指标。通俗地来讲，编辑距离指的是在两个字符串 S 和 T 之间，由其中一个字符串 S 转换为另一个字符串 T 所需要的最少单字符编辑操作次数，允许的操作有以下三种：

1. 删除 S 中的任意一个字符；
2. 在 S 中插入任意一个字符；
3. 将 S 中的一个字符替换成任意的字符。

现在给你两个字符串 $S = s_1 s_2 \cdots s_n$ 和 $T = t_1 t_2 \cdots t_m$，分别表示两个武器的字符串编码，你需要判断他们编辑距离和整数 k 的关系，并且在编辑距离小于等于 k 时输出具体的编辑方案。

Input

输入第一行包含整数 $T(1 \le T \le 10)$，表示测试数据组数。

对于每一组测试数据，第一行输入三个整数 n, m 和 $k(1 \le n, m \le 10^5, 1 \le k \le 500)$，分别表示两个字符串的长度和编辑距离需要对比的 k，接下来的两行分别输入字符串 S 和 T，保证输入的字符串仅包含小写字母。

Output

对于每一组测试数据，如果编辑距离小于等于 k，则第一行输出 “YES”，并在第二行输入他们的编辑距离 d，并在之后的 d 行中输出编辑方案：具体来说，每行输出以下三种操作之一（请注意，字符串的下标在每次操作后都会更新）：

1. “1 pos” 将字符串 S 下标为 “pos” 的字符删除；
2. “2 pos c” 向字符串 S 中插入一个字符 “c”，使其在插入后的下标为 “pos”；
3. “3 pos c” 将字符串 S 中下标为 “pos” 的字符替换为字符 “c”。

否则, 输出唯一的一行 “NO”。

Example

standard input	standard output
2 3 3 1 aab abc 3 3 3 aab abc	NO YES 2 2 4 c 1 2

Note

字符串的下标从 1 开始，并且 1 和 2 操作会使字符串的下标发生变化，在输出编辑方案时请以变化后的下标为准。

Problem E. 怪物商店

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

在海拉鲁大地上，有很多不同的怪物商店。

通过 Cuber QQ 持续不断的开垦地图，终于找齐了所有的 n 个怪物商店。对于其中的第 t 个怪物商店，里面一共售卖不同的 a_t 件物品，每个物品都有 k 个，其中第 i 种商品的售价为 i 。

现在 Cuber QQ 希望在第 t 个怪物商店，消费所买下的物品价格的平均值为一个正整数 x_t，他想知道有多少种不同的方案。

Input

输入第一行包含一个整数 $k(1 \le k \le 100)$。

第二行包含一个整数 $n(1 \le n \le 150)$，表示怪物商店的数量。

接下来的 n 行，每行两个整数 $a_t, x_t(1 \le x_t \le a_t \le 150)$ ，表示商店的属性和消费的平均价格。

Output

输出包含 n 行，每行一个整数表示方案数。由于方案数的结果可能很大，你只需要输出模 10^9+7 的结果。

Example

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

Problem F. 自建一始

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Cuber QQ 接到任务要帮助自建一始村。

一始村可以认为是一个简单无向图 $G(V,E)$，图中的顶点对应一个居民的住处，对于任意两个点 u,v，如果 $\{u,v\} \notin E$ 并且 $deg_u + deg_v \geq k$，则 Cuber QQ 可以修建这条道路，即将边 $\{u,v\}$ 加入 E 中了，其中 deg_u 表示的时图中节点 u 的度数，即顶点 u 所连边的条数。

现在 Cuber QQ 想要找到最大的 k，使得在采取最优策略时可以将给定的图补全成一个完全图，让一始村中的居民住处之间可以两两互相直接到达，完全图是指对于任意一对顶点 $u \neq v, \{u,v\} \in E$。

Input

第一行两个整数 $n,m(2 \leq n \leq 500, 0 \leq m < \frac{n(n-1)}{2})$，代表给定图的顶点数以及初始时边数。

接下来 m 行，每行两个整数 $u,v(1 \leq u,v \leq n, u \neq v)$，表示 u,v 之间有一条无向边。保证给出的图是没有重边，没有自环的简单图。

Output

输出一行一个整数，表示最大的 k，表示答案。

Example

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

Problem G. 沃托里村

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

沃托里村有一个著名的堵坊，Cuber QQ 经常在堵坊中败得倾家荡产。

这一次，Cuber QQ 发现了堵坊的规律，堵坊主有 n 个财物，他们的价值分别用整数 $a_1, a_2, \cdots, a_n$ 来表示。每次宝物存放点都和

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i \oplus a_j)^2$$

是息息相关的。$x \oplus y$ 表示的是异或运算，是按位的不进位加法，即对于 x 和 y 在二进制下的相同对应位置 a 和 b，如果 a 、b 两个值不相同，则异或结果为 1 ；如果 a 、b 两个值相同，异或结果为 0。

现在 Cuber QQ 只需要根据堵坊主的 $a_1, a_2, \cdots, a_n$ 计算出

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i \oplus a_j)^2$$

的结果，就能保证每次都能胜利了。

Input

第一行包含一个正整数 $n(1 \le n \le 10^5)$，表示数列的个数。

第二行包含 n 个正整数 $a_1, a_2, \cdots, a_n(1 \le a_i \le 10^9)$。

Output

输出一个整数，表示结果模 $10^9 + 7$ 的结果。

Examples

standard input	standard output
3 1 2 3	28
7 8 9 4 2 5 6 9	4088

Problem H. 王国之泪

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

Cuber QQ 终于玩上塞尔达王国之泪了，他欣喜地在初始天空岛上发现了一个扭蛋机。

通过研究 Cuber QQ 发现了海拉鲁大地上的 n 个不同的扭蛋机，同时他发现每个扭蛋机的属性都是不同的。具体来说，扭蛋机最多可以扭出 m 个不同的装置（编号从 1 到 m），而其中编号为 i 的扭蛋机有一个属性 $a_i(1 \le a_i \le m)$，表示这个扭蛋机每一次都会等概率的随机出一个编号从 1 到 a_i 之间的装置。

现在 Cuber QQ 想获得所有的 m 个装置，他想知道在策略最优的情况下，他需要使用扭蛋机的期望次数，毕竟左纳乌的材料也不是可以直接获得的。

Input

输入第一行包含一个整数 $n(1 \le n \le 10^6)$，表示扭蛋机的数量。

第二行包含 n 个用空格隔开的整数 $a_1, a_2, \cdots, a_n(1 \le a_i \le 10^7)$，表示扭蛋机的属性，其中 $m = \max_{i=1}^{n} a_i$。

Output

输出一个整数表示在策略最优情况下的期望的次数，由于这个数可能很大，你只需要输出模 $10^9 + 7$ 的结果。

Example

standard input	standard output
5 1 2 3 4 5	900000015

Problem I. 究极之手

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Cuber QQ 在学习了强连通集后，他自己提出了一个新的概念强不连通集。一个强不连通集是一个图中一些两两不连通的顶点所形成的集合。换句话说，强不连通集 S 由图中若干顶点组成，且 S 中任两个顶点之间没有路径。

现在 Cuber QQ 得到一个带权树，即树上的边都有权值，同时 Cuber QQ 还会给你一个集合 S，他希望通过“究极之手”删除掉树中的一些边，使得 S 成为一个强不连通集，当然，Cuber QQ 希望移除掉的树边之和最小。

Input

第一行一个整数 $n(1 \le n \le 10^6)$，表示树的大小。

接下来的 $n-1$ 行，每行三个整数 $u_i, v_i, w_i (1 \le u_i, v_i \le n, u_i \ne v_i, 1 \le w_i \le 10^6)$，表示一条树上的边 (u_i, v_i) 且权值为 w_i。

接下来的 n 行，每行一个整数 x_i，表示将结点 x_i 插入 S，一开始 S 为空集合，也就是当前询问的 S 为 $S = \cup_{k=1}^{i} x_k$。输入保证给出的 x_i 构成一个排列。

为避免大量读操作，本题采用非传统读入方式。对于每一个测试点，我们输入四个 int 范围内的整数：n $g1$ $b1$ $g2$，n 的含义如题目所示，而 $g1$ $b1$ $g2$ 则用于在本地生成测试数据。

我们使用 generator class 来生成数据（给出的参考程序为 C++，Java/Python 版本请从附件中下载）：

```
class generator {
  private:
    int N, G1, B1, G2;

    int get_edge() {
        static int x = 1, v = 2, o = -1;
        static const int p = 998244353;
        static const int mod = 1000000;
        ++o;
        if (o % 3 == 0) {
            x = (1ll * x * G1 + B1) % p;
            return x % (v - 1) + 1;
        } else if (o % 3 == 1)
            return v++;
        else {
            x = (1ll * x * G1 + B1) % p;
            return x % mod + 1;
        }
    }

    int get_permutation() {
        static int i = 0;
        ++i;
        return 1ll * i * G2 % N + 1;
    }

  public:
    generator(int n, int g1, int b1, int g2) {
        N = n, G1 = g1, B1 = b1, G2 = g2;
    }

    int input() {
        static int cnt = 0;
        ++cnt;
        if (cnt <= 3 * (N - 1))
            return get_edge();
        else
            return get_permutation();
    }
};
```

我们使用 **generator gen(n, g1, b1, g2);** 来实例化一个本地数据生成器，之后只需要调用 **gen.input()** 即可获得相应的输入。对于前 $3(n-1)$ 次调用，每三次调用会按顺序返回一条边对应的信息 u_i, v_i 与 w_i，而对于后 n 次调用，其 n 个返回值会构成 1 到 n 的一个排列。

Output

输出共 n 行，每行一个整数，表示对于每一个 S 询问的答案。为避免大量写操作，我们记对于 n 次询问的答案分别为 $ans_1, ans_2, \ldots, ans_n$。令 $res_i = (res_{i-1} * ans_i + ans_i^2)\%998244353$，最后输出 res_n 即可。

Example

standard input	standard output
5 22221 2953 54681	916548890

Note

样例对应的原始输入为

5

1 2 394408

1 3 891998

2 4 592492

3 5 777899

2 3 4 5 1

ans_1 到 ans_n 为

0

394408

986900

1764799

2656797

Problem J. 余料建造

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Cuber QQ 会给定一个初始长度为 n 的字符串 S 与一个初始为空的字符串 T，你可以进行 n 次操作，每次操作可以选择从 S 的头部或者尾部取出一个字符，并将其添加到 T 的尾部。

现在 Cuber QQ 想请你构造出字典序最小的 T。

Input

第一行一个整数 $n(1 \le n \le 2 \times 10^6)$，表示字符串的长度。

第二行一个字符串 S，保证 S 中只包含大小写英文字母。

Output

一行一个字符串 T，表示答案。

Example

standard input	standard output
6 ACDBCB	ABCBCD

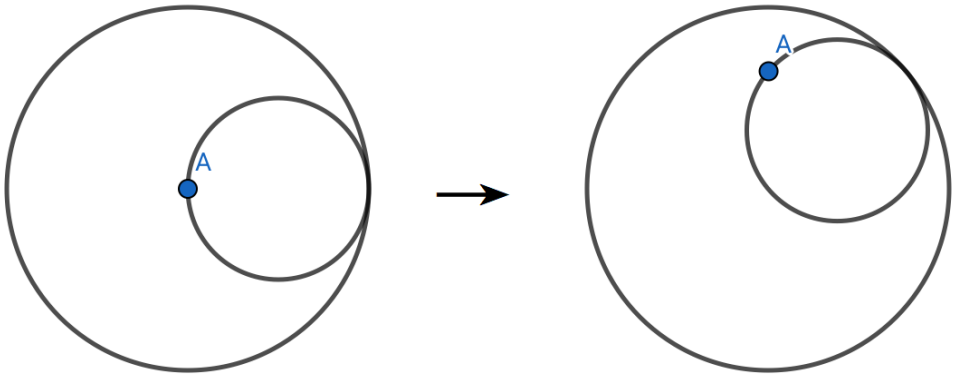
Note

对于两个字符串 $S = s_1 s_2 \dots s_n$ 以及 $T = t_1 t_2 \dots t_m$，如果在 S 和 T 从左往右第一个出现不同的位置 x 上有 $s_x < t_x$，或者 S 是 T 的前缀且 S 的长度小于 T，那么我们说 S 的字典序小于 T 的字典序。

Problem K. 倒转乾坤

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

Cuber QQ 现在手上有两个圆环，其中小圆环的直径是 d，大圆环的直径是 $2d$。他将小圆环放在大圆环内，并让小圆环紧贴大圆环内壁进行无滑动的滚动。



小圆环紧贴大圆环进行无滑动滚动时，其上一点 A 位置的变化

Cuber QQ 总是喜欢动态的美，他在小圆环上等间隔地标记了 n 个点，他想知道在小圆环贴着大圆环运动一周后，他所标记的 n 个点所经过的轨迹的长度之和是多少。

Input

输入包含一行，两个整数 $n, d (1 \le n, d \le 10^9)$ 。

Output

输出包含一行一个浮点数，表示 n 个点所经过的轨迹的长度之和，所输出的答案和标准答案的绝对误差或相对误差在 10^{-6} 范围内会被认为是正确的。

Example

standard input	standard output
1 1	4.00000000