E8 - Statement

A 与众不同的字符

题目描述

给定一个由小写英文字母组成的字符串，其中最多有一个字符是与众不同的。你能找到那个与众不同的字符吗，如果所有字符都相同，请输出 `All characters are the same`

注意，与众不同意味着它和其他字符都不同且其他字符都相同。也就是说，字符串的第 i 个字符是与众不同的，当且仅当，对于任意不大于字符串长度的正整数 j,k，如果 i,j,k 互不相同，那么「字符串的第 i 个字符与字符串的第 j 个字符不同」且「字符串的第 j 个字符与字符串的第 k 个字符相同」。

输入格式

一次输入多组数据，每组一行一个长度为 N 的字符串 S，其中 S 由小写英文字母组成，且 $3 \leq N \leq 100$。

保证 S 中最多只有一个字符与众不同。

输出格式

对于每组输入，如果字符串的第 k 个字符是与众不同的，那么输出 k；如果所有字符都相同，输出 `All characters are the same`。

样例输入

```
aaba
aaaa
```

样例输出

```
3
All characters are the same
```

B 查质数表

问题描述

给定一张包含前 N 个质数的有序质数表，请你据此判断某个数是不是质数。

输入格式

第一行输入一个正整数 N，表示将会给定前 N 个质数，保证 $1 \leq N \leq 500000$。

第二行输入 N 个正整数，表示从小到大输入的前 N 个质数，保证输入的质数不超过 10^7。

第三行输入一个正整数 t，表示接下来将查询 t 个正整数，保证 $1 \leq t \leq 500000$。

接下来 t 行，每行一个正整数，表示要查询的数字 k，保证 $1 \leq k \leq 10^7$。

输出格式

对于每次查询，输出一行，如果这个数字是给定质数表中的数字，输出 `It must be xx-th prime!`，其中 `xx` 表示它是第几个（从 1 开始）质数；如果这个数字小于给定质数表中最大的数且不在质数表中，输出 `It can't be a prime!`；如果这个数字大于给定质数表中最大的数，输出 `I'm not sure!`。

样例输入

```
5
2 3 5 7 11
4
5
10
15
17
```

样例输出

```
It must be 3-th prime!
It can't be a prime!
I'm not sure!
I'm not sure!
```

样例解释

- 5 在给定的质数表中的第三个元素，所以是第三个质数
- 10 小于质数表中最大的质数，且不在质数表中，所以一定不是质数
- 15 和 17 都大于质数表中最大的质数，所以我们无法判断它们是不是质数

Hint

除了自己实现一个二分查找外，你也可以使用 `bsearch`。

`bsearch` 是 `<stdlib.h>` 中的库函数，用于在已排序的一段数组中二分查找：

```
void *bsearch(const void *key, const void *base, size_t num, size_t size, int (*cmp)(const void *, const void *));
```

其中 `key` 为指向待查找数据的 **指针**，`base` 为待查找数组的首地址（数组名），`num` 为数组中元素个数，`size` 为单个元素的大小（一般用 `sizeof(...)` 表示），`cmp` 为自定义判断函数，定义方式如下：

```
int cmp(const void *a,const void *b) {
    if(*b 为 目标元素) return 0;
    if(*b 位于 目标元素 右侧) return 一个负整数;
    if(*b 位于 目标元素 左侧) return 一个正整数;
}
```

比如在这道题中，我们可以定义 `cmp` 为：

```
// 想一想为什么可以如此定义
int cmp(const void *a, const void *b) {
        // 无类型指针，使用前必须进行类型转换
        return *(int *)a - *(int *)b;
}
```

以下是该题一个可参考的代码片段：

```
int cmp(const void *a, const void *b) {
    return *(int *)a - *(int *)b;
}
int primes[500005];

int main() {
    // 读入 n
    // 从 0 开始读入 primes 数组
    // 读入 t
    for(int i = 1; i <= t ;i++) {
        int target;
        scanf("%d", &target);

        // bsearch 返回值为 void*，即无类型指针，接收时我们利用了一个 int* 进行了类型转换
        // 如果元素在数组中，那么 bsearch 将返回指向数组中与 key 相等的元素的指针（如果有多个相等的元素将返回任意一个）
        // 如果元素不在数组中，那么 bsearch 将返回 NULL
        int *ret = bsearch(&target, primes, n, sizeof(int), cmp);
        // 根据返回值进行相应的处理
    }
}
```

C 钢琴大师

题目描述

热爱钢琴的 shtog 偶然获得了一项特殊的能力：可以用意念操纵琴键，也就是说他可以用意念控制钢琴上任意个键被同时按下！为了熟练这项能力，他决定拿一个比较特殊的只有 32 个键（编号从左到右为 31 到 0）的智能钢琴练习一下。于是，在练习最开始时其用意念控制了若干个琴键被按下，接下来其智能钢琴屏幕上迅速闪过一系列操作，shtog 需要连贯无误地完成这一系列操作。每次操作是下列中的一种：

操作码	操作参数	含义
1	x_1,x_2	将钢琴的第 x_1 个键和第 x_2 个键（从第 0 个键开始计数，下同）的按压状态交换
2	x	如果第 x 个琴键正被按下，那么将其松开；否则，将其按下。
3	x	让第 x 个琴键处于被按下的状态
4	x	让第 x 个琴键处于被松开的状态
5	$x_1, x_2 (x_1 \leq x_2)$	让从第 x_1 个琴键到第x_2个琴键的状态逆序（比如有3个琴键依次是压下，松开，松开的状态， 将其逆序后即为松开，松开，压下的状态）

要求输出钢琴各个琴键最终的状态。

输入

第一行一个整数 $T(1 \leq T \leq 100)$，表示数据组数。

接下来 T 组数据。

每组数据第一行一个 `unsigned int` 整数 a，其 32 个位表示钢琴的 32 个键的起始按压状态（1 表示按下，0 表示松开）。第二行一个整数 t，表示该组数据一共有 t 次操作, 接下来 t 行每行一个操作，每个操作由操作码和操作的琴键索引组成，这些数字以一个空格隔开，如：`1 2 3`表示操作码为1，将第2个和第3个琴键的按压状态交换；`3 2`表示操作码为3，让第2个琴键保持被按下的状态。

输出

每组数据对应输出一行输出，输出一个无符号整数，其 32 个位表示经过这些次操作后钢琴的各个琴键的状态。

输入样例

```
2
3279105849
2
2 3
2 1
2994330020
2
1 26 0
1 22 13
```

输出样例

```
3279105843
2990143908
```

AUTHOR：shtog

D 磁盘调度

问题描述

请你编程模拟一种经典的磁盘调度算法——**SSTF（Shortest Seek Time First）**，即**最短寻道时间优先算法**。算法的过程如下：

- 初始时，磁头停在某磁道上，磁头需要访问一个磁道请求队列 $P = \{p_1, p_2, \cdots, p_n\}$
- 每次磁头都会从 P 中选择一个距离磁头当前所在磁道 q 最近的磁道 p_i（如果 P 中有多个距离 q 最近的磁道，选择在位置上最左面的一个，如 10 和 12 都离 11 最近，我们在其中选择 10），将 p_i 从 P 中删除，然后移动距离 $|q - p_i|$ 后移动到磁道 p_i
- 重复上一步直到 P 为空

现在，你需要实现一个程序模拟上面的磁盘调度算法，请结合样例理解算法的过程。

输入格式

第一行输入两个正整数 n 和 m，分别表示磁道请求的数量和磁头的初始位置，保证 $1 \leq n \leq 100$，$1 \leq m \leq 10000$。

第二行 n 个整数，表示每次请求所请求的磁道号，保证在 $[1, 10000]$ 之间，且互不相同。

输出格式

第一行输出 n 个整数，表示按处理顺序输出的请求磁道号，相邻两个整数之间用空格隔开。

第二行一个整数，表示磁头按 SSTF 算法处理完 n 个请求移动的总距离。

样例输入

```
5 50
82 170 43 140 24
```

样例输出

```
43 24 82 140 170
172
```

样例解释

- 初始时，磁头在 50 位置，有 5 个请求要处理，离磁头初始位置最近的位置是 43
- 磁头移动 7 个距离后到达 43，离磁头最近的请求位置是 24
- 磁头移动 19 个距离后到达 24，离磁头最近的请求位置是 82
- 磁头移动 58 个距离后到达 82，离磁头最近的请求位置是 140
- 磁头移动 58 个距离后到达 140，离磁头最近的位置是 170
- 磁头移动 30 个距离后到达 170，所有请求处理完毕

综上，处理顺序为 `43 24 82 140 170`，移动的总距离为 172。

E big a+b problem

题目背景

Orch1d 刚刚学会用程序解决 $a+b$ 问题，但他发现自己的程序有时仍会出错，你能根据他给出的数据范围求出正确答案吗?

题目描述

输入形式为 $a+b=$ 的表达式，其中 a,b 均为非负整数，求出 $a+b$ 的值。

输入

一行字符串，表示要计算的表达式。

输出

$a+b$ 的值

输入样例1

```
1+1=
```

输出样例1

```
2
```

输入样例2

```
99999999999999999+1=
```

输出样例2

```
100000000000000000
```

数据范围

对于30%的数据，保证 $0 \leq a,b \leq 10^{18}$。

对于100%的数据，保证 $0 \leq a,b \leq 10^{10086}$。

F 让我们一起来学习离散化

题目背景-离散化

简介

离散化是指将无限空间中的有限个体映射到有限的空间中去，以此提高算法的时空效率。通俗地说，离散化是在不改变数据相对大小的条件下，对数据进行相应的缩小。

在本题中，会将数据压缩为从 1 开始的一个排列。（具体含义见下方法）

常见运用

当有些数据因为本身很大或者类型不支持，自身无法作为数组的下标来方便地处理，而影响最终结果的只有元素之间的相对大小关系时，我们可以将原来的数据按照排名来处理问题，即离散化。

用来离散化的可以是大整数、浮点数、字符串等等。

方法（其中一个）：

通常原数组中会有重复的元素，一般把相同的元素离散化为相同的数据。

方法如下：

1. 创建原数组的副本。
2. 将副本中的值进行排序。（本题中请按照从小到大排序）
3. 将排序好的副本去重。
4. 查找原数组的每一个元素在副本中的位置，位置即为排名，将其作为离散化后的值。（本题中位置从 1 开始）

以上方法来自于oi.wiki

题目描述

下面将给一个长度为 n 的数组，分别为 $a_1, a_2, \ldots\ldots a_n$，请将这个数组**离散化**。

本题将按照上述离散化方式进行评测，如果用其他离散化方式进行离散的同学请仔细确认最后得到的结果是否与上述方法得到的结果相同。

输入格式

输入共两行。

第一行输入一个整数 n。其中，$1 \leq n \leq 3 \times 10^5$。

第二行输入 n 个整数，分别为 $a_1, a_2, \ldots\ldots, a_n$。其中，$-10^9 \leq a_i \leq 10^9$。

输出格式

输出共 1 行。

第 i 个输出的数为 a_i **离散化**后的结果，每两个数之间用一个空格做间隔。

样例输入

```
10
3 9 7 8 2 6 10 7 9 7
```

样例输出

```
2 6 4 5 1 3 7 4 6 4
```

样例解释

步骤二结束后副本为：$2,3,6,7,7,7,8,9,9,10$

步骤三结束后副本为：$2,3,6,7,8,9,10$

因而原数组中 2 **离散化**后为 1， 3 **离散化**后为 2，其他数以此类推。

HINT

如何排序更快呢？如何查找更快呢？

Author：小P

G 弦论

题目描述

dyc酷爱上了**弦论**！

为了更深地研究这一理论，他想要搞明白这样一个问题：对于一个给定的字符串，它的字典序第 k 小的子串是什么？（注意：不同位置的相同字串算作多个）

如果你不知道子串的含义，那么dyc告诉你：子串是字符串中任意个连续的字符组成的字符串。

输入

第一行是一个仅由小写英文字母构成的字符串 s，保证 $|s| \leq 100$ 。

第二行一个正整数 k , k 的含义如题目所示，保证 $1 \leq k \leq 10^9$。

输出

输出数据仅有一行,该行有一个字符串，为第 k 小的子串。若子串数目不足 k 个，则输出 `-1` 。

输入样例

```
aabc
3
```

输出样例

```
aa
```

HINT

字符串字典序比较可以使用 `strcmp` 函数。

如果字符串的长度更长，有没有更好的方法呢？

H 圣诞老人送礼物

题目描述

圣诞节要结束了，圣诞老人骑着他的驯鹿从天而降来到了最后一座需要送礼物的城市，这座城市比较特殊，其中的楼房都排成了一排，楼房的高度可能不同，圣诞老人会将给某栋楼的居民的礼物送到楼顶让大家自行领取。

由于驯鹿这两天工作很累了，所以这次送礼物的过程中它始终**只能**往**更低的方向**飞（而不能向着**同样高度**或**更高**的方向飞）以保证体力够用，同时掉头也会耗费很多体力，所以驯鹿在送礼物的过程中只能**始终向左边飞**或者**始终向右边飞**而**不能转向**（驯鹿和圣诞老人具有神奇的魔力，可以以透明状态穿过大楼，也就是说在向更低的楼层飞行时，不会被中间的比较高的楼挡住去路）。在保证驯鹿体力够用的前提下，善良的圣诞老人希望给尽可能多的楼房送礼物，初始时他们可以随机降落到任何一座楼的楼顶，请编写程序计算出圣诞老人最多能给几栋楼送上礼物。

输入

第一行一个整数 $T(1 \leq T \leq 50)$，表示有T组数据

每组数据有两行输入：

第一行有一个整数为楼房的个数 $n(1 \leq n \leq 1000)$。

第二行 n 个以单个空格分隔开的整数（末尾没有多余的空格），表示 n 个楼房各自的高度 $h(1 \leq h \leq 10000)$（按照楼房一条线上排列的顺序按序给出）。

输出

输出一共 T 行，每行一个整数，第i行的整数表示第i组数据下圣诞老人最多能送到礼物的楼房数。

输入样例

```
2
3
1 3 2
4
4 3 6 9
```

输出样例

```
2
3
```

对于第一组数据，应该从左往右飞 `3,2` 或者从右往左飞 `3,1`

对于第二组数据，应该从右往左飞 `9,6,3`

AUTHOR：shtog

I 格雷码？格雷码！

注意本题的时间与空间限制

题目描述

Gino 在写一道算法题时，由于时间复杂度太高而 `TLE` 了。好在他发现可以用格雷码来降低一层时间复杂度！

格雷码（Gray Code）是一种特殊的 n 位二进制串排列法，它要求相邻的两个二进制串间恰好有一位不同，特别地，第一个串与最后一个串也算作相邻。

所有 2 位二进制串按格雷码排列的一个例子为：00，01，11，10。

n 位格雷码不止一种，下面给出其中一种格雷码的生成算法：

1. 1 位格雷码由两个 1 位二进制串组成，顺序为：0，1。
2. $n+1$ 位格雷码的前 2^n 个二进制串，可以由依此算法生成的 n 位格雷码（总共 2^n 个 n 位二进制串）按**顺序**排列，再在每个串前加一个前缀 0 构成。
3. $n+1$ 位格雷码的后 2^n 个二进制串，可以由依此算法生成的 n 位格雷码（总共 2^n 个 n 位二进制串）按**逆序**排列，再在每个串前加一个前缀 1 构成。

综上，$n+1$ 位格雷码，由 n 位格雷码的 2^n 个二进制串按顺序排列再加前缀 0，和按逆序排列再加前缀 1 构成，共 2^{n+1} 个二进制串。另外，对于 n 位格雷码中的 2^n 个二进制串，我们按上述算法得到的排列顺序将它们从 $0 \sim 2^n-1$ 编号。

按该算法，2 位格雷码可以这样推出：

1. 已知 1 位格雷码为 0，1。
2. 前两个格雷码为 00，01。后两个格雷码为 11，10。合并得到 00，01，11，10，编号依次为 0 ~ 3。

同理，3 位格雷码可以这样推出：

1. 已知 2 位格雷码为：00，01，11，10。
2. 前四个格雷码为：000，001，011，010。后四个格雷码为：110，111，101，100。合并得到：000，001，011，010，110，111，101，100，编号依次为 0 ~ 7。

现给出 n, l, r，Gino 想得到按照上述算法生成的 n 位格雷码中，第 l 到第 r 号之间所有的二进制串。

输入

一行，三个非负整数 n, l, r，含义与题意相同，保证 $1 \le n \le 60, 0 \le l < r < 2^n, 1 \le r-l \le 5 \times 10^7$。

输出

为了减小输出量，只需输出以下内容：

第一行输出一个 n 位的二进制串，表示编号为 l 的 n 位格雷码。

随后，将每一个编号对应的 n 位二进制串 s 都看成其对应的无符号整数形式 v（例如，111 看作 7，010 看作 2），在第二行输出

$$\left(\sum_{i=l}^{r} v_i \times 13331^{i-l}\right) \bmod 2^{64}$$

注：对 2^{64} 取模等同于使用 `unsigned long long` 计算自然溢出后的结果。

输入样例 1

```
3 5 7
```

输出样例 1

```
111
710928906
```

输入样例 2

```
24 114514 1919810
```

输出样例 2

```
00000001011000000111111011
3616982174085117875
```

样例解释

对于第一组样例，三位格雷码为 000，001，011，010，110，111，101，100，编号为 $5 \sim 7$ 之间的为 111，101，100，对应的无符号整数为 $7, 5, 4$，$7 + 5 \times 13331 + 4 \times 13331^2 = 710928906$

Author: Gino

J 可分数列

题目背景

设 m 为正整数，数列 $a_1, a_2, \ldots, a_{4m+2}$ 是公差不为 0 的等差数列，若从中删去两项 a_i 和 a_j $(i < j)$ 后剩余的 $4m$ 项可被平均分成 m 组，且每组的 4 个数都能够成等差数列，则称数列 $a_1, a_2, \ldots, a_{4m+2}$ 是 $(i,\ j)-$ 可分数列。

题目描述

已知 m, i, j 的值，求一个可行方案，将剩余的 $4m$ 项可被平均分成 m 组，且每组的 4 个数都能够成等差数列。

输入

多组输入，每组一行 3 个正整数，分别为 m, i, j，含义与题目描述相同。

保证 $1 \leq i < j \leq 4m + 2 \leq 10^6$，且输入不超过 5 组。

（为与题目中「平均分成 m 组」区分，此后输入输出的「多组」改称「多轮」）

输出

对于每轮 m, i, j，对应的输出为：

- 若有可行方案，输出 m 行，即原数列分成的 m 组每组一行，输出其在原数列中的 **下标**，同组的 4 个数之间用一个空格间隔。
- 若无可行方案，输出一行 `Nonseparable!`。

本题采用 `Special Judge`：有多种可行方案时输出任意一种即可；每轮中分成的 m 组输出顺序不做要求；但每组中 4 个数必须按递增或递减顺序输出。

输入样例

```
1 3 4
6 10 13
```

输出样例

```
Nonseparable!
1 9 17 25
2 4 6 8
3 7 11 15
5 12 19 26
14 16 18 20
24 23 22 21
```

HINT

从 $1, 2, \ldots, 4m+2$ 中一次任取两个数 i 和 j $(i<j)$，记数列 $a_1, a_2, \ldots, a_{4m+2}$ 是 $(i,\ j)-$ 可分数列的概率为 P_m，则 $P_8 = \frac{27}{187} > \frac{1}{8}$。