

哈尔滨工业大学(威海)

2018年新生ACM程序设计竞赛

问题集

时间：2018.12.23 13:00 - 17:00 封榜时间：16:00

比赛地址：<http://10.245.130.74/>

问题编号	问题名	页码
A	Arcaea	4
B	Battlestations Pacific	6
C	Cities: Skylines	8
D	Dragon Quest	10
E	Ever17 -the out of infinity-	12
F	Fire Emblem	14
G	Grand Theft Auto	17
H	Hacknet	19

注意事项

1. 问题的时间与空间限制在每道题目的页面上，时空限制为单点时空限制，而非整体时空限制。
2. 你可以使用页面右侧的 Clarification 系统向我们提问，我们的公告也会发布在此。你可以前往 Problemset 页面下载完整问题集以及每道题目的所有样例输入和输出。
3. 题目难度不随题号递增而递增。请自己寻找哪些题是你可以做的。
4. 请仔细阅读下一页开始的样例问题后再开始答题。我们不会回答任何样例问题内提到的问题！
5. 你应该提交 .c , .cpp , .java 等源代码文件，而不是 .exe , .cbp 等文件。
6. 使用 C/C++ 的选手，main 函数最后一定记得要 return 0 ; main 函数的形式应该为 int main ，而不是 void main 或者无返回值。
7. 使用 Java 的选手，请确保你的主类名字是 Main。由于 Java 语言特性问题，你的时空限制会被稍微放大。
8. 比赛时间为 13:00 - 17:00，在 16:00 榜单会被冻结：新的提交仍然计数，但是不会显示在榜单上。
9. 罚时采用标准 ICPC 规则：第一次正确提交作为本题的通过时间，除 COMPILER ERROR 之外的错误提交计 20 分钟罚时，COMPILER ERROR 不计罚时。通过后的所有提交均不计算罚时。
10. Good Luck and Have Fun! Hope you all bugless codes!

提交反馈

- PENDING : 你已经成功提交，正在评测队列中等待评测。
- CORRECT : 恭喜！你的程序在我们期望的时空限制内运行结束而且答案完全正确！
- WRONG-ANSWER : 你的程序在时空限制内运行结束，但是答案错误。
- COMPILER-ERROR : 你的程序编译错误。记得检查你是不是语言选错了。
- TIMELIMIT : 你的程序没有在期望的时间内运行结束。
- MEMORY-LIMIT : 你的程序申请了多于限制的内存。
- RUN-ERROR : 你的程序出现了运行错误，如除 0，数组访问越界等。
- TOO-LATE : 比赛结束啦~ 你的提交已经不会被计算在结果内。

X. 样例问题

时间限制	空间限制
1000 ms	256 MB

题目描述

你好！

本问题并不是比赛问题或者训练问题，而是为了向你解释一般的问题格式而特制的样例问题。所以，本题没有提交入口！

本问题的题目描述如下：

这是一个非常简单的问题。请你输出 $A + B$ 的值。

一个问题由如下部分构成：

1. 问题编号与名称：认准问题编号和名称再提交才能不出错！
2. 时间限制和空间限制：一个问题有多个测试点，我们取所有测试点中运行时间最长，使用内存最多的成绩作为你的提交成绩，并不是总计。要注意的是，运行时间最长和使用内存最多的测试点，并不总是同一个测试点。只有你的提交成绩在时空限制之内并且给出正确答案的，我们才看做你正确。
3. 题目描述：描述了问题的背景，问题的定义和问题的内容等。仔细阅读这一部分理解问题，之后设计算法并实现才能通过题目。
4. 输入格式：描述了我们会输入什么，输入的限制是什么，以及按什么格式输入。请注意，我们保证输入严格按照我们所描述的规则递交给你的程序，所以请不要使用任何 `printf("请输入一个大于0小于233的整数T:")` 这类提示语句！这会导致你获得一次 Wrong Answer！
5. 输出格式：描述了你应该按照什么样的格式输出你的答案。请注意，一定要保证严格按照我们要求的格式输出，否则你会获得一次 Wrong Answer；另外，我们的评测输入输出分离，并不会如你测试一样混合，所以请你不要使用任何诸如 `system("clear")` 之类的系统调用！这可能会导致你获得一次无意义的 Runtime Error！
6. 样例：样例有两部分：标准输入流代表我们会给你程序什么，标准输出流代表当使用标准输入流给出的输入时，你的程序应该输出什么。请注意，样例通常不会包含所有情况，所以并不是样例过了就过了。过了样例不一定能过本题，样例都没过一定过不了本题，所以请先使用样例进行测试再提交！
7. 提示：包含了出题人给你的友好提示，包括但不限于怎么解释样例等：)。

当你了解了每一个部分的意义后，阅读一下本题（尤其是后面的样例代码）。如果没有问题了，那就翻到两页后，开始做题吧！

输入格式

输入仅包含一行，为两个整数 A 和 B ($1 \leq A, B \leq 100$)。

输出格式

输出仅包含一行，为 $A + B$ 的值。

样例

标准输入流	标准输出流
12	3
45	9

提示

下列代码将通过本题。

C :

```
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d\n", a + b);
    return 0;
}
```

C++ :

```
#include <iostream>

using namespace std;

int main()
{
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

Java :

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a, b;
        a = sc.nextInt();
        b = sc.nextInt();
        System.out.println(a + b);
    }
}
```

A. Arcaea

时间限制	空间限制
1500 ms	64 MB

题目描述

你喜欢玩音游吗？Ramen 可是一位音游达人。最近，他又发现了一个好玩的音游，叫做 Arcaea。



在 Arcaea 的世界里面，我们使用潜力值系统（Potential Point, 简称 ptt, 下同）来评定玩家的技术实力，ptt 越高，玩家的技术水平也就越高。

Ramen 对这个系统是如何运作的很感兴趣，经过一顿操作（也叫作百度）之后，他发现 ptt 是这么计算的：

首先，一首歌有一个**定数**，这个定数就是官方评定的这首歌的难易水平计算基准。然后，当玩家打出一个成绩以后，我们通过这样的方式来计算玩家的单曲 ptt（分数记为 s ，定数记为 p ）：

分数	ptt
$s \geq 10,000,000$	$p + 2$
$10,000,000 > s \geq 9,950,000$	$p + 1.5 + \frac{s - 9,950,000}{100,000}$
$9,950,000 > s \geq 9,800,000$	$p + 1 + \frac{s - 9,800,000}{400,000}$
$9,800,000 > s$	$p + \frac{s - 9,500,000}{300,000}$

要注意，单曲 ptt 最低是 0。而当我们得出单曲 ptt 以后，我们采用如下的办法计算玩家的当前总 ptt：前 30 个最好成绩加上最近 10 次的成绩的 ptt 之和除以 40。这里还有一条特殊的规则：如果某个成绩已经参与了前 30 个最好成绩的计算，那么它就不参加最近 10 次成绩的统计了。举例来说，如果玩家的倒数第 2 个成绩参加了最好的前 30 个成绩的计算，那么在计算最近的 10 次成绩的时候就不再使用这条成绩进行计算。

现在，Ramen 使用了特殊方法（也叫作查 wiki）获得了他最近打的每首歌的定数，他也记下了他当前打歌的 ptt。你能帮他写一个小工具，让他能快速计算他的 ptt，进而更好的提升自己的实力吗？

输入格式

输入包含多组数据。

输入的第一行为一个整数 $T(1 \leq T \leq 100)$ ，代表测试用例的组数。

接下来的 T 组测试用例按照如下格式给出：

第一行为一个整数 $N(40 \leq N \leq 1000)$ ，代表成绩条数。

接下来的 N 行，每行包含两个数 $s, p(0 \leq s \leq 10000000, 1.0 \leq p \leq 12.0)$ ，代表每首歌的成绩和定数。 p 仅包含一个小数位。

给出成绩的顺序为由上到下由新到老。

输出格式

对于每一组测试用例，在新的一行中输出该成绩组的总 ptt。特别的，请把这个数四舍五入至 2 位小数。

样例

注意，由于本题输入输出较多，不在题面提供样例，请大家在问题集页面自行下载样例输入输出！

提示

保留两位小数可以直接使用 `printf` 来四舍五入，不需要自己判断。

ptt 的计算方法题目有简化，实际上比这个更复杂，如有兴趣请参考：<http://wiki.arcaea.cn/index.php?title=Potential>

B. Battlestations Pacific

时间限制	空间限制
1000 ms	64 MB

题目描述



海战，是二战中最具浪漫色彩的部分之一，也是二战史中最为闪光的一个篇章。作为忠实的海战迷，SRC 最大的乐趣就是在 Battlestations Pacific 这款游戏中还原历史战役的情景。（SRC：历史梗天下第一！）

在海战历史上，有这么一场神奇的战役，我们值得一谈。指挥官 Mr. Kidding 拥有一个很大的舰队，和一个不靠谱的传令官。这场战役最神奇的地方，就在 Kidding 指挥官的战列编排上。

我们的 Kidding 指挥官有两种舰群：巡洋舰（Cruisers，使用 C 表示）和驱逐舰（Destroyer，使用 D 表示）。一开始，Kidding 的战列只有一艘驱逐舰。然后，Kidding 指挥官使用如下办法布置他的战列：

- 按当前战列配置一组完全相同的舰队；
- 把新战列的每一艘船对换，也就是说，如果当前战列某个位置的船只为巡洋舰，那就换成驱逐舰，反之亦然；
- 然后，再把新战列的顺序翻转。比如，原来是 DDC，翻转过后就变成了 CDD。
- 最后，把新的战列阵容拼在之前的战列之后。

当然，这样配置的舰队还不够安全，所以 Kidding 指挥官还会在两个战列之间添加一艘驱逐舰（D）保证两个战列的安全。Kidding 指挥官会进行多轮这样的操作，直到他觉得当前战列足够令人满意为止。

我们用字符串来表示的话，Kidding 的战列变化历程是如下样子的：

$$\begin{aligned}S_1 &= D \\S_2 &= DDC \\S_3 &= DDCDDCC \\\dots\end{aligned}$$

现在，你已经知道 Kidding 指挥官是如何配置战列的了，现在就该不靠谱的传令官出场了！排列战列是一项很频繁的举动，Kidding 指挥官总是下各种指令，而我们优秀的传令官总是会在各条指令间辗转，终于——Kidding 指挥官有了一个 S_J 状态的战列阵容！Whoa！什么，你问 J 是什么？当然是 Joke 常数啦！顺带一提， $J = 233^{233}$ 。

天道好轮回，苍天饶过谁。这么大的舰队，可怜的传令官根本算不清楚哪里是什么船，而 Kidding 指挥官的命令还在不断下达。SRC 看不过去了，准备帮助这位可怜的传令官给 Kidding 指挥官答案。

毕竟，SRC 在游戏中，就是控制着 Kidding 指挥官：)

输入格式

输入包含多行。

第一行为一个整数 $T(1 \leq T \leq 23333)$ ，代表测试用例的组数。

接下来的 T 行，每一行包含一个整数 $K(1 \leq K \leq 2.33 \times 10^{18})$ ，表示 Kidding 指挥官想要询问的位置。

输出格式

对于每一个测试用例，在新的一行中输出对应位置的舰船类型。巡洋舰为 Cruisers，驱逐舰为 Destroyer。

样例

标准输入流	标准输出流
3	Destroyer
2	Cruisers
6	Cruisers
14	

提示

考虑前 20 艘船：DDCDDCCDDCCDCCDDCD.....，可以看出第 14 艘船是巡洋舰。

这场战役并不存在。

C. Cities: Skylines

时间限制	空间限制
1000 ms	64 MB

题目描述

你好！市长！



你说你一个好好的程序员，怎么就当上市长了呢？

当上市长了，自然要好好规划你的都市，这就是你的第二个家。当然，为了满足市民们的各种需要，建设一些新的基础设施是有必要的——花一点钱又何妨？市民的笑脸远比那点钱来的温暖的多。

建筑之间往往不是独立的，两个建筑 A, B 之间自然会对各自产生影响。我们定义，两个建筑的总和贡献是两个建筑的乘积。然而，这个定义还有一个小小的缺陷：两个都是负贡献的建筑，合在一起就变成正的了？怎么可能？当然是越来越差！所以，我们把两个建筑的贡献，定义为运算 \otimes ，它的运算规则如下：

- $a \otimes b = a \times b$
- $(-a) \otimes b = -(a \times b)$
- $a \otimes (-b) = -(a \times b)$
- $(-a) \otimes (-b) = -(a \times b)$

最近由于预算充足，你决定修一部分新的建筑来丰富你的城市。你的智囊团已经为你评估好了每两个建筑的影响，但是他们之中没有人懂如何计算两个建筑的总和贡献。作为市长，以及一位（前）优秀程序员，这样的计算怎会难得倒你？赶快完成，后面还有一堆文件等着签呢！

输入格式

输入包含多组数据。

输入的第一行为一个整数 T ($1 \leq T \leq 100$)，代表测试用例的组数。

接下来的 T 行，每一行有两个整数 n, m ($|n|, |m| \leq 10^9$)，代表两个城市的贡献值。

输出格式

对于每一组测试用例，在新的一行中两个城市的总和贡献。

样例

标准输入流	标准输出流
4	1
1 1	-1
-1 1	-1
1 -1	-1
-1 -1	-1

提示

请使用 long long 来保存答案。如果使用 scanf 和 printf，请使用 "%lld"。

D. Dragon Quest

时间限制	空间限制
1000 ms	64 MB

题目描述



RPG , Role Playing Game , 我们所熟悉的角色扮演游戏。这个世界上有许多十分知名的 RPG 游戏 , 而熟悉 RPG 的玩家 , 一定会听说过勇者斗恶龙这个传奇的游戏系列。黄金三人组的开发阵容 , 永远不变的幻想故事 , 历久弥新的回合制战斗 , 是这个游戏仍然有如此多拥趸的秘诀。

你是勇者斗恶龙系列的粉丝 , 现在正在玩最新的《勇者斗恶龙11》。由于你带错了装备 , 所以当前你的小队如果直接攻击这个 BOSS , 每回合只能造成 1 的伤害。不过 , 你有一个技能 , 可以给这个 BOSS 一次性造成 K 的伤害。这个技能的副作用是 , 这个角色在下一回合会变得暂时不可用 , 必须等一个回合才能继续使用。由于这个 BOSS 血量充足 , 你很快就败下阵来。你翻了翻攻略 , 如果需要找到能够给这个 BOSS 一定伤害的武器 , 你需要做一系列支线任务——不仅枯燥 , 而且需要很长的时间。所以你想硬干 , 当然 , 无谋的硬干只是浪费时间 , 所以你想知道 , 有多少种不同的方法打败当前 BOSS ?

举例来说 , 假如技能伤害为 3 , 那么攻击一个血量为 4 的 BOSS 有以下三种方法 :

1. 连续直接攻击 BOSS。
2. 先直接攻击一次 BOSS , 然后使用一次技能。
3. 先使用一次技能 , 在下一个回合由于技能不可用 , 所以你直接攻击 BOSS。

注意 , 这个 BOSS 还有另一个技能 , 就是如果你当前造成的伤害超过了 BOSS 的生命值 , 他会把多出来的这部分反馈到你身上 , 所以 , 你只考虑正好击败 BOSS 的方法。

我们定义两种攻击方式中 , 假如有任何一个时刻你所使用的战斗方法不同 , 这两种攻击方式就是不同的。

由于答案可能会很大 , 你只需要输出对 2^{64} 取模后的结果即可。取模运算在 C/C++/Java 里面使用 % 操作符。

输入格式

输入包含多行。

第一行为一个整数 T ($1 \leq T \leq 233$)，代表测试用例的组数。

接下来的 T 行，每行包含两个整数 K_i, D_i ，分别代表技能的攻击力和 BOSS 的血量。

数据范围满足 $1 \leq D_i \leq 10^5, 2 \leq K_i \leq 10^5, 1 \leq i \leq T$ 。

输出格式

对于每个测试用例，在新的一行中输出正好击败 BOSS 的方法数对 2^{64} 取余的结果。

样例

标准输入流	标准输出流
3	3
3 4	5
3 6	15
11 23	

提示

如果你不知道如何把结果对 2^{64} 取余的话，请你使用 `unsigned long long` 来保存中间变量和结果，在自然溢出的状态下就是对 2^{64} 取模。

E. Ever17 -the out of infinity-

时间限制	空间限制
1000 ms	64 MB

题目背景



如果说世界上有什么不可思议的话，你应该是确实遇到了：你只是来新开的水下游乐园 LeMU 享受难得的周末，为什么就被困在了这里？

等到回过神来，你发现自己站在了一扇密码门前。你想起来了，你们一行人在被困后找到了一个据点，商量好对策之后，你的任务是尽可能去远处探索，找一下有什么补给，或者有没有什么逃生的手段。毕竟，距离 LeMU 崩坏，只剩下不到 60 个小时了。

你定了定神，看了看这扇门。这扇门和乐园的氛围格格不入——倒不如说，这扇门更应该出现在医院或者实验室之类的地方。你定睛一看，门的密码键盘上面有这么一张纸条，似乎是粗心的研究人员用于提醒自己密码是什么而写的：

求下面多项式的所有零点：

$$f(x) = P_N \cdot x^N + P_{N-1} \cdot x^{N-1} + \cdots + P_1 \cdot x^1 + P_0 \cdot x^0$$

当你即将开始计算的时候，小町鵠（游戏女主角之一）指了指下面，你看到了另一句话：

该多项式的运算优先级翻转。也就是说，加法 + 优先级最高，乘法 \cdot , \times 次之，乘幂运算 x^k 最低。

例如， $a^b + c \cdot d = a^{(b+c) \cdot d}$ 。特别的， $0^0 = 1$ 。

你傻傻的笑了一下，刚让她对你的好感度上升，这里却不能露怯。只是，这扇门背后到底隐藏了什么？一想到这里，你计算速度就开始变慢。想让这种和谐的时光变得更长，甚至，要不就多呆一会儿？反正距离 LeMU 崩坏还剩五十个小时……

“你到底在干什么？”“啊哈哈……”思绪被她的一声质问拉了回来。不管门背后是什么，你都要为了未来而前进！你加快了手上的计算，答案就快出来了，不能让她再多等了，也不能让大家再多等了！

但是，这次事件背后所隐藏着的真相，远超过你的想象。我是指，正在玩这个游戏的你。

输入格式

输入包含多行。

第一行为一个整数 $T (1 \leq T \leq 233)$ ，代表测试用例的组数。

接下来有 $2T$ 行，每两行按如下格式给出一个测试用例：

第一行为一个整数 $N (0 \leq N \leq 233)$ ，代表这个多项式的项数。

接下来的一行有 $N + 1$ 个整数 $|P_i| \leq 233, 0 \leq i \leq N$ ，代表这个多项式第 i 项的系数，即系数按照 x 升幂顺序给出。我们保证 $P_N \neq 0$ 。

输出格式

对于每个测试用例，在第一行中输出解的个数。如果有无穷多个解，请输出 INF，如果没有解，请输出 0。

如果有解且有有限个解，则在接下来的一行中按任意顺序输出所有的解，用空格分隔。

如果解的个数正确，同时每个解与正确答案之间的绝对误差或者相对误差在 10^{-6} 以内，都会被认为是正确的。我们保证只有实数解。

样例

标准输入流	标准输出流
3	1
1	0.0
1 1	0
4	0
9 0 -6 2 -2	
0	
19	

提示

这个例子也许能帮助你理解题目中的运算优先级： $-2^{-3} \times -1 + -2 = -2^{(-3 \times (-1 + -2))} = -2^9 = -512$ 。

如果对运算优先级仍有问题，请通过 Clarification 提问。

第二个多项式为：

$$f(x) = (-2)x^4 + 2x^3 + (-6)x^2 + 0x^1 + 9x^0$$

按照题意展开，发现此式无零点。

F. Fire Emblem

时间限制	空间限制
8000 ms	384 MB

题目描述

如果要提到战棋类游戏，火焰纹章（Fire Emblem）系列一定是不得不提的。



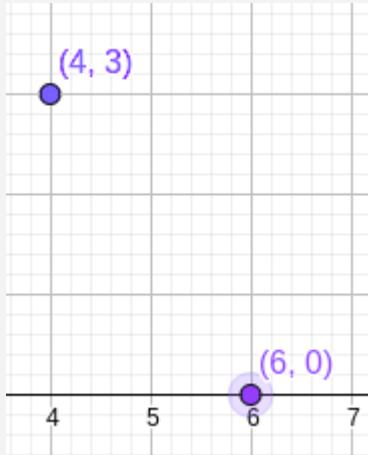
让我来简单描述一下游戏规则吧。游戏的规则很简单，你有一些可操作单位和一些敌人在一个矩形网格上，每个单位的装备啊，可移动范围啊等等都不同。双方按回合轮流移动各自单位，每个单位都有自己的移动规则，一旦到达规则限制了本回合就不能再移动了。你要在本队不被全灭 / 重要单位不被击败等限制条件下，完成击败掉全部敌人或者防御限定回合等要求。

有一点不得不提：一个单位的移动方向只能是上下左右。换句话说，一个单位在网格上的移动路径都是由有限条直线组成的。对于任意两个单位来说，由于我们只能上下左右移动，所以这两个单位的距离是曼哈顿距离。

两个单位的曼哈顿距离是什么？简单来说，对于两个点 $(x_1, y_1), (x_2, y_2)$ ，他们的曼哈顿距离

$$D_m = |x_1 - x_2| + |y_1 - y_2|。$$

比如，下图的两个点对的曼哈顿距离为 5。



那么，两个单位的最短曼哈顿距离越大，意味着我可能需要更多的回合才能让两个单位尽可能靠近，他们互相支援的难度也就越大。让我们来考虑一个问题：对于你现有的单位来说，互相支援的难度有多大？

让我把问题形式化描述一下：假设当前你有一张 $N \times N$ 大小的正方形地图，每一格都有一些单位 C 。那么，我知道的问题其实是：假设两个单位之间没有阻碍，可以随意移动，那么对于所有的两个单位来说，他们之间的最短曼哈顿距离是多少？当然，这样的问题太大了，而且并无益处，所以我只想知道，对于我任意选择的两个单位，有多少种可能的曼哈顿距离？每种可能的曼哈顿距离又有多少对？

可以相信的是，如果能解出这个问题，你就能成为火焰纹章的高玩。事不宜迟，赶快行动吧！

输入格式

输入包含多行。

第一行为一个整数 $N (1 \leq N \leq 1024)$ ，代表网格图的大小。

接下来有 N 行，每行有 N 个数字，按如下形式给出：

$$\begin{matrix} C_{1,1} & \cdots & C_{1,N} \\ \vdots & \ddots & \vdots \\ C_{N,1} & \cdots & C_{N,N} \end{matrix}$$

$C_{i,j}$ 代表在 (i, j) 的位置有 $C_{i,j}$ 个单位， $0 \leq C_{i,j} \leq 10, \forall i, j, 1 \leq i, j \leq N$ 。

输出格式

输出包含多行。

第一行为一个整数 K ，代表有 K 种距离。

接下来的 K 行，每行各两个整数 $D_i, Q_i (1 \leq i \leq K)$ ，代表距离为 D_i 的点对有 Q_i 对。

请按 D_i 由上到下升序的顺序输出。

样例

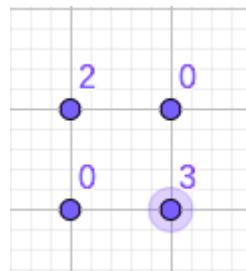
标准输入流	标准输出流
2 20 03	2 04 26
3 123 456 789	5 0120 1340 2340 3160 430

提示

对于样例一来说，我们有如下两种距离：

- $D_1 = 0$ ，在位置 $(1, 1)$ 的有 $C_2^2 = 1$ 对，在位置 $(2, 2)$ 有 $C_3^2 = 3$ 对，共有 $1 + 3 = 4$ 对。
- $D_2 = 2$ ，我们可以选择任意一个 $(1, 1)$ 的单位和任意一个 $(2, 2)$ 的单位配对，所以答案是 $2 \times 3 = 6$ 。

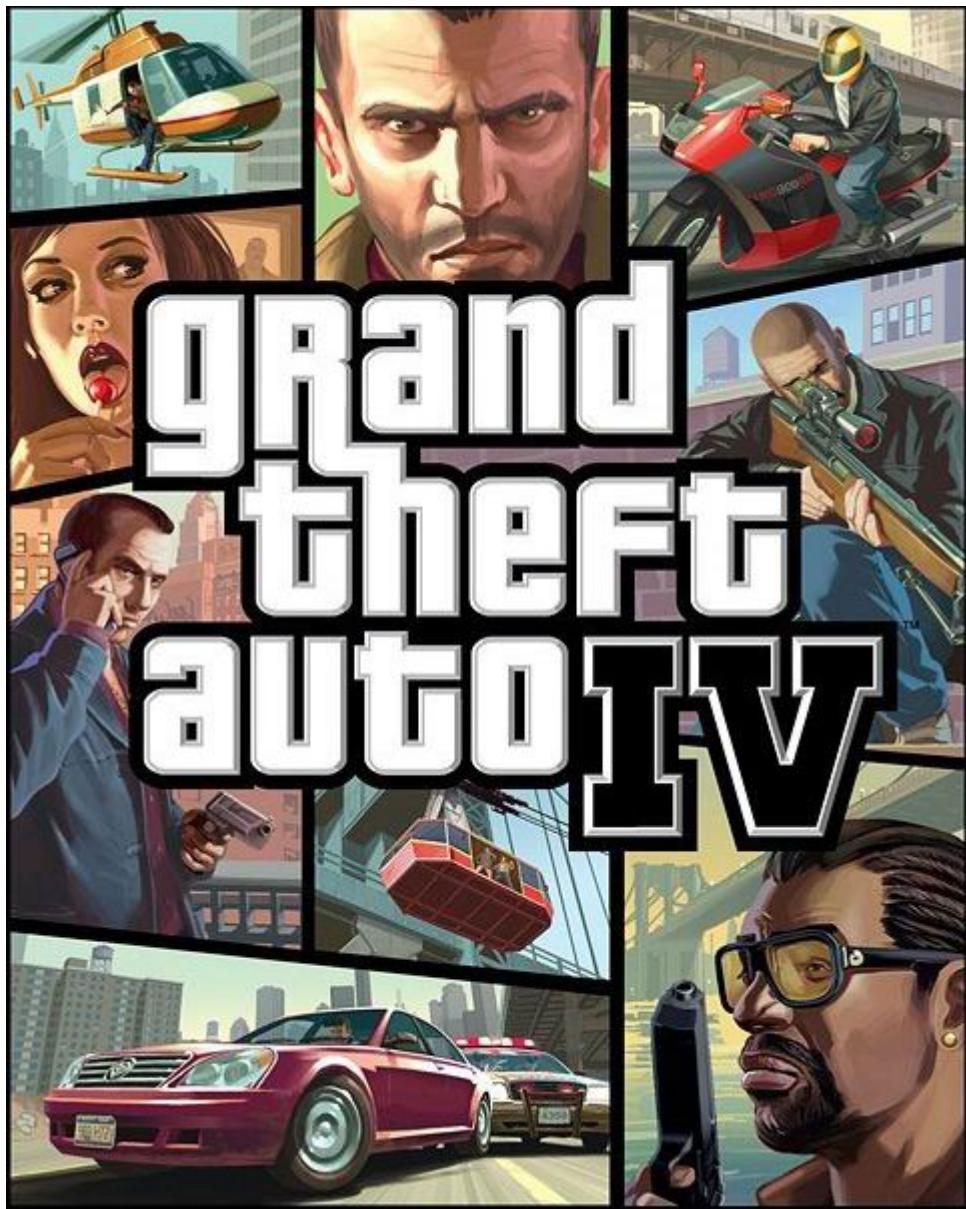
下面的图片可能帮助你更好的理解：



G. Grand Theft Auto

时间限制	空间限制
1000 ms	64 MB

题目描述



自由城，造就了这个全世界人口最密集、最充满活力的城市中心。这里既是梦想兴起者的天堂，也是梦想衰败者的地狱。

你，Niko，经历了重重考验，已经在自由城站稳了脚跟。作为当前黑帮的主力——其实也不是什么主力，就只是一个小头头而已——你也有自己的小弟，也有自己可以调动的“兵力”和资源。

自由城，机会繁多，从来不是一个安静的地方。今天，隔壁街区的黑帮又来挑事，而火拼的地方正好有两个在你的领域里面，你可以每一个地点派一个小队过去。在太岁头上动土？你们还嫩着点！

然而，战斗是要顾全大局的，如果你们这战斗力太强，吸引了周围的火力，就你目前的阵容来说是根本招架不住的；相反，如果战斗力太弱，你肯定是在这个地方混不下去了。怎么办？幸好，你还有一个算靠谱的表哥——脑子还算可以，他告诉你，大概满足 N 的火力值，就是一个很好的选择。

“这个 N 又是一个什么东西？”

“听好了。对于一个团体来说，我们有进攻火力值 A ：什么枪啊刀啊棒球棍啊，还有一个防御值 D ，也就是比如防爆盾和跑路用的摩托车什么的。对于你的两个团体，他们能够给你的整体火力值，可以用如下方程计算出来：

$$N = \min\{|A_a - D_a|, |A_b - D_b|\} \oplus A_a \oplus D_a \oplus A_b \oplus D_b$$

其中 $\min\{x\}$ ，代表取 x 中的最小值， \oplus 代表异或运算， $|x|$ 代表取 x 的绝对值。你只要能够合理的分配你的火力和防御就行了。”

那这是你的工作了。你的火力和防御还不少，你想放多少都没有问题；怎么计算火力值和防御值呢？没问题，你有表哥！所以，开始你的部署吧，战斗一触即发了！

输入格式

输入包含多行。

第一行为一个整数 $T(1 \leq T \leq 23333)$ ，代表测试例的组数。

接下来的 T 行，每一行包含一个整数 $N(0 \leq N \leq 2.33 \times 10^8)$ ，代表目标火力值。

输出格式

对于每个测试用例，在新的一行中 4 个整数 A_a, D_a, A_b, D_b ，代表目标两个团体各自的火力值和防御值，要求 $0 \leq A_a, D_a, A_b, D_b \leq 10^9$ 。

如果找不到这样的答案，请你输出 -1 。如果有多个答案，输出任何一组都是可以的。

样例

标准输入流	标准输出流
3	3 1 2 6
4	1 1 2 2
0	7 9 6 5
12	

提示

在 C/C++/Java 中，异或运算使用操作符 \wedge 。

对于第二组样例，还有很多可行答案，其中一个 $0 0 0 0$ 。

下面这个关于异或的运算律可能对你会有一定帮助： $x \oplus 0 = x, x \oplus x = 0$ 。

H. Hacknet

时间限制	空间限制
1000 ms	64 MB

题目描述

- > I need your help
- > I have everything you'll need
- > I will teach you how to use it
- > In exchange, I want you to find me
- > My name is ... Bit
- > And if you're reading this
- > I'm already **dead**



这或许不是什么好事。或者说，这就不是一件好事。

当你从邮箱看到这个视频的时候，你知道，阴谋找上了你。当然，谁心中没有一个英雄梦？梦想着哪天能凭借自己的努力，去拯救一些人，去做一些惊天动地的事。你也是这么想的，所以你欣然应允了。Bit 没有违背他的承诺，他不仅给你留下了一条完整的黑客成长路线，为你介绍了进入知名黑客组织的方式，为你提供了养活自己的工作，让你能够接触这么一个神奇的世界。今天之前，你都会认为这是一件天大的好事，这是你是天选之子的证明！

.....如果没发生这件事的话。

闲话休谈。当你深入这件事之后，你发现 Bit 死的真相并没有那么单纯。这件事的确是一件彻头彻尾的阴谋，很多看似不可能的碎片连了起来，就如同这台老旧服务器居然是保存了 Bit 和神秘人交易记录一样，这是你完全没有想到的。你同样没有想到的是，这台老旧服务器居然有一个埋得这么深的钩子，你只是进去下载了一份聊天记录，居然从层层代理节点中追踪到了你。幸运的是，这个钩子虽然很难破解，但是在黑客组织的资源服务器上，你找到了一个这样的函数，能够帮助你解决这个钩子：

```

long long func(long long x)
{
    if (x < 20190001)
        return x + 2018;
    else
        return func(func(x - 2019));
}

```

通过一定手段，你能获得当前这个钩子的状态 x ，把 x 当参数调用这个函数，你就可以获得使这个钩子停止工作的密码。

你看了看右侧的网络热力图，现在大概有数以万计的肉鸡服务器向你发起了攻击。同时，深谙社会工程学的你，知道你的 ip 已经暴露，后果不容设想。如果还想保护自己，你必须要在几秒内让这个钩子停下来。事不宜迟，现在就开始防御！

然而，这段代码的最后修改时间距现在已经有点年头了，你很怀疑它能不能正常工作。毕竟，留给你的时间并不多。

输入格式

输入包含多行。

第一行为一个整数 $T(1 \leq T \leq 10)$ ，代表函数状态的个数。

接下来的 T 行，每一行包含一个整数 $x(1 \leq x \leq 10^{18})$ ，代表每一个函数的状态。

输出格式

对于每一个测试用例，在新的一行中输出对应函数状态下使这个钩子停止工作的密码。

样例

标准输入流	标准输出流
2 1 20192018	2019 20192018

提示

请使用 `long long` 保存输入和答案。如果使用 `scanf/printf`，请使用 `%lld`。

如果你的提交获得了 `TIMELIMIT` 或者 `RUN-ERROR` 的反馈，请不要尝试重复提交，再下一次提交前请仔细思考。

必要的时候，你可以重写这个函数。我们保证这个函数可以计算出答案且答案唯一。

Java 版本的代码如下：

```

public long func(long x)
{
    if (x < 20190001)
        return x + 2018;
    else
        return func(func(x - 2019));
}

```