April 6, 2025

2025 年洛谷愚人节比赛 题解

2025 愚人节团队@Luogu

TOC

【OI】

A / 每日打卡心情好

By cff_0102

输入的第一行是没用的，因为完全无法确定它是以什么格式给出的。直接把它 `getline` 掉。

实际上，只需要在输入第一行后 `while(cin>>x)` 即可做到完整读取后面的内容。

但是如果直接输出最后连续 1 的个数只能拿 40pts，因为洛谷的打卡机制不是这样的。

题面的最后一句话提示我们去题库中找和打卡有关的题目。不难找到 P1664，它的题目名称与本题一致。

在这道题中详细介绍了洛谷的打卡机制，直接按题意模拟即可。

本题的 #6 测试点特意 hack 了一些直接使用位运算的代码。好像还有不少理解错了部分原题题意的，因为数据有点水被放到了 50~60pts。~~因此建议先到 P1664 交过一遍后再来写这题~~。

B / 连号子序列数

By cff_0102

诈骗题。

考虑每个 1 到 N 之间的连续区间对答案产生多少贡献，不难发现对于每个连续区间，都可以在排列中找到唯一一个子序列与之对应，即贡献为 1。因此答案就是连续区间的数量，即 $\frac{N(N+1)}{2}$。

C++ 要开 `long long`。

C / LGR-S 2025 括号序列

By ffzero

还是诈骗题。

知道吗，轴对称也是可以上下对称的，而 texttt 的 $\{*\}$ 这三个字符都是上下对称的，所以任何字符串都是轴对称的……答案就是……用 3^n 减去原题的答案。

D / 关键的钥匙

By E.Space

这道题主要需要明白这个关键的钥匙是什么，以及怎么用。

题面说题目背景的图片告诉了你如何使用这个钥匙。这是一张 100×1000 的图片，包含黑色和透明两部分。

输入格式是一个很长的数字串。

这道题还有英文题面，内容就是中文翻译了一遍。

所有的线索指向了键盘。输入格式、图片的比例、图片中的透明部分告诉我们需要关注键盘上 1 到 0 的数字键，而英文题面则告诉我们，钥匙 = key = 键。

图片中透明的部分标记了左边五个键的下半部分和右边五个键的上半部分，也就是说，我们只需要使用这些上半部分和下半部分。通常来说，数字键的下半部分是数字，上半部分是符号（不过有些键盘上不会这么画）。

所以说，这个“关键的钥匙”应该是 shift 键！也就是，把输入中的 `67890` 分别转换为 `^&*()` 然后对表达式求值即可。运算符的优先级是和 C++、Python 相同的 `*` 高于 `&` 高于 `^`。通过样例可以验证这一猜想的正确性。此外样例输入的内容都是误导项。

注意可能有特别大的数字。如果你使用 `unsigned int` 可以获得 42 分。当然，Python 可以很轻松地解决这个问题。下面的代码可以得到满分（注意要交 PyPy 3 语言）：

```
s=input()
s=s.replace('6','^')
s=s.replace('7','&')
s=s.replace('8','*')
s=s.replace('9','(')
s=s.replace('0',')')
print(exec(s))
```

感谢 Eason_cyx 造的数据。

E / 旷野小计算

By ffzero

- Case 1：数乘

略微思考就能发现 $4080 = 4096 - 16 = 2^{12} - 2^4$。

```
^ y x
< y 12
< x 4
- y x
! x
```

- Case 2：数除

首先，下取整操作有个性质：$\frac{\frac{x}{a}}{b}$ 等于 $\frac{\frac{x}{b}}{a}$。

所以，对 $\frac{x}{c}$ 简单变形，可以取一个合适的常数 k，变为 $\frac{x\times 2^k}{c\times 2^k}$，然后提取出 $\frac{2^k}{c}$ 记作 q，于是原式变为 $x \times \frac{q}{2^k}$，也就是计算 $x \times q$ 然后右移 k 位。这里的 q 是可以预先根据 c 计算出来的常数。

那么，k 取多少合适呢？可以尽量大，只要保证 $x \times q$ 不溢出 64 位即可。x 是 32 位的，所以只要保证 $\frac{2^k}{c}$ 不超过 32 位就行。在本 case，k 取 43。

其实这就是编译器会做的事：把常数除法转化为更快的常数乘法。一个偷懒的办法：去 godbolt，编译 `x / 4080`，就能得到常数 q 了。

- Case 3：归一

本题并没有提供常数节点。那要怎么获得常数呢？首先需要一个稳定的常数 1 的来源，有了 1 就可以通过位移和异或得到其他常数。

对于本 case，一个简单的想法是：分离x 的 64 位比特，然后用一个累加器做按位或运算。本题也没有支持跳转的节点，所以需要手动循环展开写 64 次代码，但是指令数会超长。

真的不好讲怎么想到这个做法的，但是这个做法确实有效：对于任意 $x < 2^{63}$，$0 - x$ 会因为溢出，最高位变为 1。此时只要取最高位即可。

但是对于 $x > 2^{63}$ 呢？取相反数后最高位会变为 0。不过不难注意到，x 的最高位已经是 1 了，所以可以得到一个解法：取 x 和 $-x$ 的最高位，相加。

这个解法很接近正解了，只是它在 $x = 2^{63}$ 的时候会有小问题：相加后的结果是 2 不是 1。解决办法有很多，而且都不难，可以取末尾两个比特再次相加，或者是再次取相反数然后拿最高位，等等。

- Case 4：量乘

乘法可以转化为加法，逻辑类似于龟速乘。用 C 语言表述出来是这样的：

```
a = 0;
for (int i = 0; i < 63; i++ ) {
    if (y & (1 << i)) a += x << i;
}
```

循环可以通过展开实现。按位与可以通过加法和异或实现。但是，`if` 语句怎么实现呢？需要放开思路，把分支结构转为顺序结构——也就是变成一个表达式来计算。首先，发现可以转化为：

```
a += (x << i) * BIT(y, i); // BIT(y, i) 表示取 y 二进制下第 i 位
```

问题转化为动态计算 $v \times b$，其中 $b = 0$ 或 1，即 0/1 乘法。也就是要找到一个操作，能够根据 b 的值让 v 不变或者 v 变成 0。

最关键的一步：发现 1 和 `0×FFFF FFFF FFFF FFFF` 并没有本质区别。后者不过是前者取个相反数。

此时答案已经很明显了：把 b 取相反数，然后与 v 进行按位与运算，答案就是 $v \times b$。总结一下，把 `if` 语句改写成：

```
a += (x << i) & -BIT(y, i);
```

有了加法运算和异或运算，不难利用性质 `a + b = 2 * (a & b) + (a ^ b)` 实现按位与运算。

至此，本 case 的所有技术障碍都已解决。

- Case 5：量除

此时的除法已经没办法转为乘法了，但是可以转为减法。也许二分法可以做，这里采用累减法，也就是列竖式计算除法。

同样，先写 C 代码再考虑翻译为指令：

```
a = 0;
for (int i = 63; i ⩾ 0; i--) {
    b = x >> i;
    if (b > y) y -= b, a ^= 1 << i;
}
```

不难想到通过相减然后取符号位来实现比大小。根据 case 4 的经验，可以把 `if` 里的表达式转化为 0/1 乘法。有 `1 << i` 这个表达式，所以需要用 case 3 的办法造出常数 1 来。

总体其实没有更难。

- Case 6：开根

比较暴力的做法是做 32 次乘法。这就是为什么允许超额一万多行。如果写出来了也会给一点分。

正解是手撕根号。具体做法的 C++ 代码：

```
c = 1;
for(int i = 31; i ⩾ 0; i--) {
```

```
    a <<= 2;
    a |= x >> 62;
    y = d << 2 | 1;
    b = a - y >> 63;
    a -= (1 - b) * y;
    d = y >> 1;
    d ^= b;
    d ^= 1;
    x <<=2 ;
}
cout<<d;
```

- Case 7：容量

本 case 是比较经典的容量问题，参考 `vector` 等一些容器的实现，会扩容到 2 的整数次幂。此时就需要快速计算出扩容后的容量。

这里直接给出 C 实现的代码。

```
x -= 1;
x |= x >> 32;
x |= x >> 16;
x |= x >> 8;
x |= x >> 4;
x |= x >> 2;
x |= x >> 1;
x += 1;
```

按位或运算可以利用性质 `a + b == (a & b) + (a | b)` 实现。略微压行即可拿到满分。

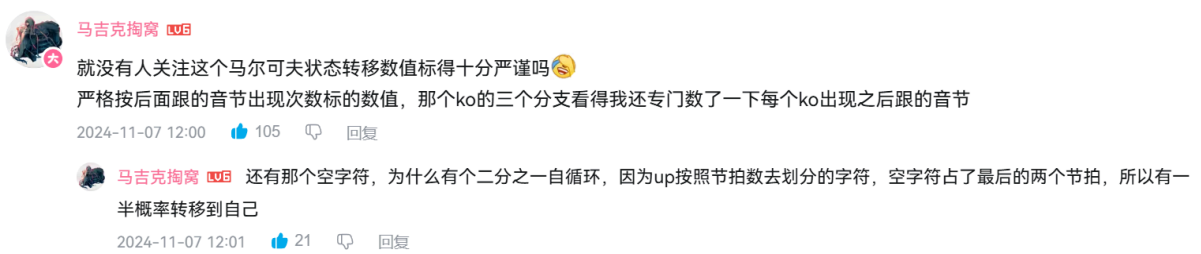
F / 指鹿为马

By E.Space

最后一组样例输入的开头是 `BV`，指向了这个视频 https://www.bilibili.com/video/BV1fy411e7ue/。看到视频标题中既有“鹿”又有“马”，看起来这个视频是有用的。

视频内容是把一个字符串转换成马尔可夫链之后不断随机采样。可以发现前几个样例的答案都等于字符串长度，其它的都比串长大一些。结合这一点，视频中的模型，视频简介，以及本题是个 OI 题的事实，从这四者之间可以合理推断，答案为从字符串首字母开始，按照马尔可夫链生成字符串，每次往末尾添加一个字符的过程中，输入字符串首次作为子串完整出现时，当前字符串长度的期望。另外我怀疑视频简介中的内容可能有正确性问题，但是与本题无关，所以没有仔细研究。

现在的问题就变成了，如何把这个输入的字符串转换为这个马尔可夫链，以及视频中的字符串是什么。这两点可以从评论区得到：



所以视频中的字符串应该是 `しかのこのこのここしたんたん`（注意后面有两个空格），以及马尔可夫链的转移概率是，统计每一种字符之后出现的下一个字符的频率，以这个频率作为当前字符转移到下一个字符的概率分布。注意最后一个字符的下一个字符是第一个字符，这个视频中也有体现。

现在的问题就变成了如何求这个期望。

我们尝试使用动态规划的方式来计算，记 $f(i)$ 为当前字符串最长的满足是【输入字符串的前缀】的后缀的长度为 i 时，期望还需要添加多少个字符才能出现完整的字符串。

于是就有

$$f(i)=1+\sum_c p_c\big(f\big(next_{i,c}\big)+\big[next_{i,c}=1\big]extra_c\big)$$

其中 $next_{i,c}$ 为当前字符串之后加上字符 c 时 i 会变成什么，可以证明这只和 i,c 有关，可以使用 KMP 算法得到。

注意当 $next_{i,c}=0$ 时，需要一直随机直到出现首字母才会让 i 变成 1。所以针对这一部分，我们设 $extra_c$ 为，当前末尾字符是 c 时，到首次出现字符串首字母结束，期望要生成多少个字符。然后定义此时的 $next$ 为 1。特殊地，对于首字母 c，$extra_c=0$。$extra$ 的值可以通过解规模等于字符集大小的线性方程组得到。

这样我们有 $f(n)=0$ 且答案为 $1+f(1)$。

但是，这个转移是有环的，我们需要解方程组。注意到等号右边一定有一项 $f(i+1)$，所以我们可以移项，使得等号的一边只有 $f(i+1)$，其余的内容在等号另一边。

然后我们可以直接设 $x=f(1)$，然后对于 $1\le i<n$，可以通过移项后的等式用 x 表示出 $f(i+1)$ 的值。可以看出 $f(i+1)$ 是之前所有 f 以及常数 1 的线性表示，所以任何 $f(i+1)$ 的值一定是一个关于 x 的一次函数。

于是最后我们会有 $f(n)=kx+b$，而又因为 $f(n)=0$，所以我们最后的答案就是 $1-\frac{b}{k}$。

算法的时间复杂度为 $O(\Sigma^3+n\Sigma)$，其中 Σ 为字符集大小。

P.S. 我在这里公布一下两个特殊性质的具体内容：

特殊性质 A：保证输入的字符串是由一个没有重复字符的字符串重复若干次得到，比如 `cat`，`catcat`，`meowmeowmeow`。

特殊性质 B：保证输入的字符串的首字母在字符串中出现恰好一次。

特殊性质 A 意味着答案恰好等于串长，也就是说直接输出串长可以得到 20 分。（貌似实际有 26 分）

特殊性质 B 意味着 $next_{i,c}$ 总是等于 1，也就是说你此时不需要使用 KMP 来计算 $next$。

当然，也许你没有分析出在不满足特殊性质 A 或特殊性质 B 时本题的题意是什么，这时你也可以得到这部分的分数。

题目中“保证没有畸形数据”的含义是，保证 k 在模 998244353 的意义下不为 0。

【PH】

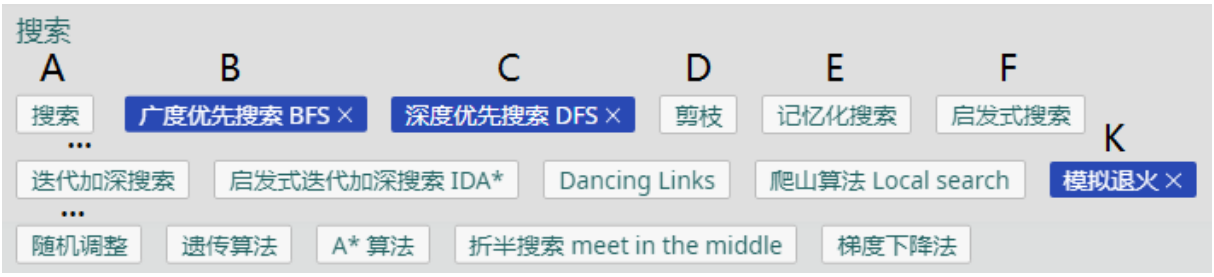
G / 算法题

By cff_0102

根据提示，要“搜到与本题算法相匹配的题目”，第一步应该要看这道题有哪些算法标签。展开即可发现本题的 10 个算法标签。

进入题目搜索页面，选择算法标签，会发现题目中给出的那十个“大类”都是这里面的算法分类，且本题的每个标签都恰好在一个分类里。但是，找到了它们的位置又有什么用呢？

提示的最后给出了五个算法以及它们对应的字母。同样在标签界面里找，会发现在同一行的算法分别位于同一个大类里，且其对应的字母在字母表中的位置恰好就是这个算法标签在这个大类中的位置：



（其中第一行的 SA 是 Suffix Array 的缩写，第二行的 SA 是 Simulated Annealing 的缩写）

猜测要对那十个标签做同样的操作，并按题面中的分号和句号进行断句。做完之后，可以提取出 `tag id of cdq`。

洛谷上 CDQ 分治的 tag id 是 100，因此输出 100 即可。

本题用到了 PH 中一个十分基础的手段：A1Z26。得到一些 ≤ 26 的数字时，不妨试试把它按字母表的顺序转成字母，那可能就是你要找的答案。

H / 谷谷谜

By BpbjsGreen

第一小题：通过组词（瞪眼法）得到第一行是“合同书”，第二行是“认同感”，所以中间的等号代表“同”。以此类推，第三行是“？同体”，容易知道答案是“共”。

第二小题：通过左边样例“云→去”“月→甩”得到箭头对应的变换是加一笔成新字。右边是“日→？→百”与“大→(？/木)→术”，得到两个问号表示的字是“白”和“犬”。

但是答案竟然是一个字！总览五道小题，后四道小题的题面都出现了“变异”，这提示我们把获得的答案也进行相应的操作。

这个小题的题面被反色了，小题答案中的白也要被反色变成黑，所以这道小题的答案是“默”。

第三小题：还原原题面后，得到黄色方块代表“四”，橙色方块代表“十”，绿色方块代表“二”。所以第三行表示“一年有二十四??”，问号处应该填“节气”。这个小题的题面被左右颠倒了，所以答案也要颠倒一下，为“气节”。

第四小题：这个小题的题面顶栏出现了三次，所以 emoji 的数量乘三，答案为“三头六臂”。

第五小题：原答案为“加沙”。这个小题的顶栏中的两个“谷”加上了衣字旁，所以答案也要加两个“衣”，为“袈裟”。

META：将五个小题的答案均按题面方向旋转，得到五个部件“并”“土”“衣”“キ”“丬”，拼装得到“拼装”。哈哈。

闲话：关注微信公众号绿谜谢谢喵。关注微信公众号咚咚谜谢谢喵。

I / 拼好题

By [cff_0102]

根据题目背景，这五题都是由三道**原题**拼凑而成的。这启发我们要找到它们分别来自哪些题目。

先简要的看一遍所有题面，里面出现了“高桥君”等字眼，并且在最后的说明提示中也提到要去找“小 A”，不难想到这些题目都是来自 AtCoder 的题目（缅怀 AtCoder RMJ qwq）。恰巧提示部分剩下的小 P，小 B，小 C，小 U，小 S 都是洛谷其它题库的名称（主题库、入门与面试、CodeForces、UVA、SPOJ），这印证了我们的想法。

洛谷题目搜索界面中有“搜索题目内容”的功能，勾选上，就能找到对应的题目。注意搜到的第一个题目不一定就是正确的，要点开搜到的前几个题目进行检查。这五个题面都是直接由原题的每一行 mix 起来的，所以不难搜到。

最终可以得到这五道题分别是由以下题目拼成的：

```
ARC025A AGC025A ABC112A
ARC178D AGC034D ABC034D
ARC139C AGC069C ABC019C
ARC153B AGC050B ABC204B
ARC138B AGC043B ABC226B
```

可以发现，每一组都含有 ARC，AGC，ABC 各一道题，且它们在原比赛中的位置是一致的（如第一组都是 A 题）。

注意到 ARC AGC ABC 可以提取出 RGB，且题目背景中说的是“三原题”而不是“三道原题”，因此这可能和颜色有关。恰好这些题目编号中的数字都小于 256，三道题可以组成一个特定的颜色。同时题目背景中提到的“一千六百七十七万多”指的就是 RGB 可以表示的颜色数量 $256^3 = 16777216$。

再看题目背景，提示我们在浏览器里面找到这些颜色的名字。经过搜索，可以发现它们的名字分别是：

```
midnightblue (25,25,112)
firebrick (178,34,34)
saddlebrown (139,69,19)
darkorchid (153,50,204)
bluevoilet (138,43,226)
```

得到名字之后怎么做呢?题目背景中提示我们“在三道原题中找到一个共同的参数”，而前面发现的“它们在原比赛中的位置是一致的”这个信息还没有用上，猜测“共同的参数”指的就是这个。

将它们对应到这些名字中并提取对应位置的字母，可以得到：

```
[m]idnightblue (A)
fir[e]brick (D)
sa[d]dlebrown (C)
d[a]rkorchid (B)
b[l]uevoilet (B)
```

拼起来就可以得到答案 `medal`。

本题用到了 PH 中一个较为常见的提取方法。在解出小问得到单词后，如果它们存在一个对应的数字，不妨试试提取单词中数字对应位置的字母，拼起来，那可能就是你要找的答案。如果没有那个数字，也可以试试提取第一个字母，或者沿对角线提取（第 i 个单词提取第 i 个字母）等方法。

J / 意义不明的图片

By tiger2005

首先需要注意到，这一系列图片来自曾经比较热门的“派对鹦鹉”系列表情包。你可以通过搜索“Party or Die”找到这些表情包。

在派对鹦鹉的官网中提供了表情包的下载链接，在下载后发现存在 HD 表情和普通表情，根据最后的描述，应当只保留 HD 表情。接下来考虑比对前两行：

```
vacationparrot        │
unicornparrot         ├⟶ vueparrot
evilparrot            │

redenvelopeparrot     │
everythingsfineparrot │
accessibleparrot      ├⟶ reactparrot
ceilingparrot         │
thumbsupparrot        │
```

通过观察可以得到本题的规律：通过缩略图比对找到表情的文件名称，随后提取首字母即可。接下来给出剩余三行的解密结果：

```
SUBMIT
PHPARROT
GIF
```

那么对于三个问题，需要满足的条件分别为：

- 提交任意一个 GIF，这可以通过检查文件头信息判断；
- 提交 `phparrot` 字符串，大小写均可；
- 提交 HD 文件夹下的 `phparrot.gif` 文件。

因为 Typst 没法渲染动图，所以这里就不扔派对鹦鹉 GIF 了。

K / Bingo?

By cff_0102

本题某些小问一定程度上受到 CCBC15 外推法 的启发。

本题给出了若干个 Bingo 的结果，每个 Bingo 的盘面一样，都是 $01 \sim 25$ 的数字。你需要根据最终得到的图案猜测这是什么东西对应的结果。小问的标题后面带有答案长度的提示。

第一问，选中的数是 $02, 03, 05, 07, 11, 13, 17, 19, 23$，它们都是质数，因此答案是 `prime`。

第二问，选中的数是 $04, 08, 12, 16, 21, 24$，除了 21 以外都是 4 的倍数，而什么和 4 有关的东西从 20 变成了 21 呢？不难联想到 2020 年东京奥运会推迟到了 2021 年举办。因此答案为 `olympics`。

第三问，选中的数是 $03, 11, 14, 17, 20, 23$，从 11 开始每两项之间的差均为 3。同时根据标题“Revision”的提示，不难联想到 C++11，C++14，……恰好是每三年更新一次。因为要提交纯英文，所以答案为 `cpp`。

第四问，出现了不同的颜色。搜索标题，可以搜到棋盘密码，这是一种可以将 26 个字母映射到 5×5 的表格上的加密手段，且在 PH 中也不少见。$01, 09, 11, 15, 17$ 的位置对应的字母分别是 a, i/j, l, p, r。一行五个字母，字母可以标成黄色和绿色，可以联想到 Wordle。最终根据 Wordle 的局面可以得到答案是 `april`。其实在得出五个字母后，如果你注意力够好，直接将它们重排（anagram）就可以得到答案。

第五问，出现了更多颜色。可以发现第一位数字相同的背景颜色是相同的，第二位数字相同的文字颜色是相同的（除了 $11, 22$ 外）。可以想到 cmd 中的 color 命令，经过尝试确实是对应的，且 `color 11` 和 `color 22` 是无效操作。因为标题是 Color，所以答案应该是 `cmd`，字数也是对应的。

第六问，标题提示我们把数字 $\times 10$。在不同范围内的数字有不同的颜色，分别是灰、蓝、绿、橙、红，可以联想到洛谷的名字颜色。经过查询可以发现这幅图就是不同咕值对应的名字颜色。因此答案为 `guzhi`。

第七问，变回了只有一种颜色的情况。标题翻译出来是“十进制”，而选中的只有 $01, 02, 10, 11, 12, 20, 21, 22$，恰好都可以看成三进制数，因此答案为三进制的英文 `ternary`。

第八问，标题提示这是二月的某个东西，那么数字应该就是日期了。在洛谷里到处找找，可以发现洛谷在这些日期中都有公开赛举办。因此答案为 `contest`。

P.S. 本题是在二月前出完的，本来我以为二月的比赛基本都会出现在周末，对应一下就可以想到答案，没想到二月比赛安排这么混乱。原本想改成其它月的，但是当时不知道为什么把 2 月 4 号选上了，导致形成了轴对称图形。轴对称也太巧了吧！于是干脆不改了，这一问就这样留了下来。

第九问，标题提示这是摩斯密码。直接翻译 ... / - / .- / .. / -. 就可以得到答案 `stain`。

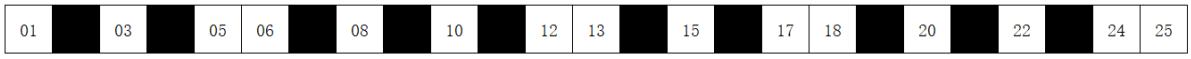
第十问，搜索标题可以搜到 C++ 的一个函数 `std::adjacent_difference()`，可以很方便的计算差分数组。把 $1, 9, 14, 15, 19$ 差分一下得到 $1, 8, 5, 1, 4$，通过 A1Z26 转换成字母就是答案 `ahead`。

第十一问，标题是去年愚人节比赛的最后一题。提取题目图片中的钟表上对应时间的字母，拼起来即可得到答案 `most`。

第十二问，标题是著名数学家康威。其实浏览器给了很大的提示：搜索康威，搜索引擎会推给你一堆和生命游戏有关的内容。正好这张图片就是生命游戏中一个著名的图案，答案是其英文 `spaceship`。

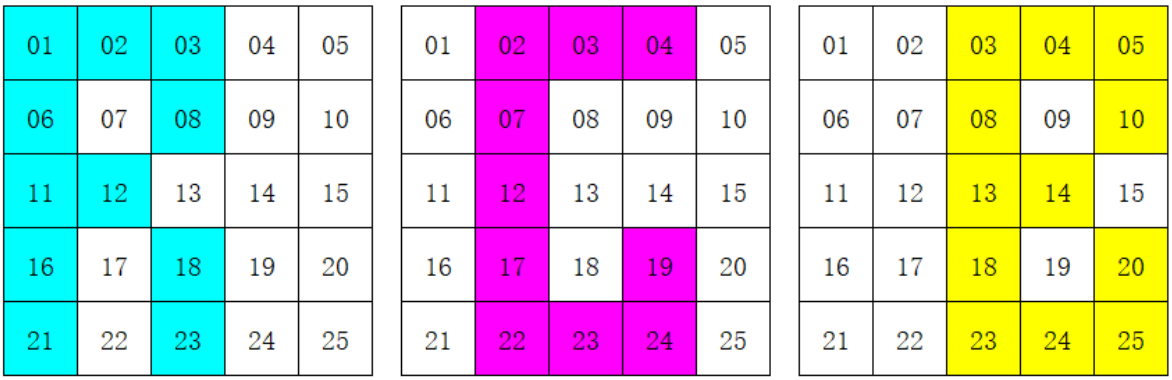
第十三问，标题是一个叫“数织”的纸笔谜题。因为图案已经画出来了，反推原来的数字，得到横纵的数字都是 $1, 4, 4, 5, 4$。A1Z26 即可得到答案 `added`。

第十四问，标题提示我们把它转化成长度为 25 的黑白序列。直接把它们拼起来：



可以发现这个序列与钢琴的黑白键顺序相匹配，恰好跨了两个八度。因此答案为 `piano`。

第十五问，标题是颜料三原色（加上黑色）的缩写。根据提示把每个格子需要的 C(Cyan), M(Magenta), Y(Yellow) 三个颜色通道分离开来，可以得到：



象形得到答案是 `rgb`。

最终答案：

```
prime
olympics
cpp
april
cmd
guzhi
ternary
contest
stain
ahead
most
spaceship
added
piano
rgb
```

L / Bingo!

By cff_0102

PH 区怎么能没有 Meta 呢?

之前听出题组其他成员说要给这个比赛出到 26 题，我就想利用这个出个 Meta 性质的题目了。

当时想的是，除去这个 Meta，剩下 25 题，刚好可以组成 5×5 的 Bingo 盘面，在 Bingo? 已经出好的前提下，再搞一个叫 Bingo! 的 Meta 题，表示这个题用的是真正的 Bingo 机制：盘面上是 25 道题目，给出若干个描述，在盘面上选上符合描述的题目，看对应描述是否在盘面上连成了线，是则提取 1，不是则提取 0，接着再对二进制进一步转化到答案。发到出题群里，得到了认可。

于是就开始等题目数量凑到 24 或 25。

当题目数量凑到 23，可以开始继续思考 Meta 怎么出了，并初步凑一点描述出来。这时我又想到了一个看起来更好的 Meta 方式：凑 25 个描述出来组成一个 Bingo 盘面，对每个题目做一次这个 Bingo，看看有没有连成线再提取为二进制。这样就不需要一定凑到 25 题了！于是开始想哪些可以作为本题的答案。

但是 $24 \sim 26$ 位二进制可以表示的东西还是太少了，没想到什么合适的答案。

此时我想到 Bingo 不一定只能连出一条线，如果把连出几条线提取出来，就可以不是二进制了，能表示的东西也就更多了。

注意到三位三进制可以表示 $0 \sim 26$ 的数，于是想到出 24 题，得到长度为 24 的三进制串，三位一断再转化成字母，答案就可以是长度为 8 的单词或词组了。

又想了一堆长度为 8 的答案，最后选中了 `accepted` 这个 2 并不算多，且很有意义的单词作为答案。

接着就是凑描述了。我前前后后想了 40 种左右不同的描述，凑了两天，最后终于在尽量少修改题面的前提下凑出来一个可以满足要求的盘面。同时在一个没有用到的格子里面放了一个本题提取方式的提示。

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijk
10000010000100101010001000000111100100  0   AGLOQSWde              | OSW    0         0    dfg    0    Qe     0    AGLj
10000010001100101010011000001001110010  0   AGKLOQSVWbe            | KOSW   0   b     0    Vfg    0    Qe     0    AGLj
10000000000100111011011010000111111110  11  ALOPQSTVWYde           | OSW    11        11   [Vdfg] 0    Qeh    0    ALPTYij
10000000001100101110010100001001111100  0   AKLOQRSVXbe            | KOS    0   b     0    Vfg    0    Qeh    0    ALRXi
10001001101110101010011000001011111110  11  AEHIKLMOQSVWbde        | KOSW   11  b     11   [Vdfg] 0    Qeh    0    AEHILMij
10000000000100111010101000000001110110  0   ALOPQSUWe              | OSW    0   U     0    fg     0    Qe     0    ALPij
00100010000000010000001100011100111011  0   CGOVWabc               | OW     0   bc    0    Vfg    0    hk     0    CGaj
00100000001000101110010000001001111101  11  CKOQRSVbe              | KOS    11  b     11   Vfg    1    [Qehk] 0    CRi
00100010001000100000011000111011111011  0   CGKOVWabce             | OW     0   bc    0    Vfg    0    ehk    0    CGKaj
00101000100000101110000100010101101 01  0   CEIOQRSXbd             | OS     0   b     0    dfg    0    Qk     0    CEIRXi
00100100011000100110000010001011101 01  11  CFJKORSYce             | KOS    11  Fc    11   fg     1    ek     1    CJRYi
00100100011001100100 11?0100100111101  222 CFJKNORUVYbc(W) N 必选| KO(W)  222 [FUbc] 11   Vfg    1    hk     1    CJNRYi
01000100001010101010001000000111011111  11  BFKMOQSWde             | KOSW   11  F     11   dg     1    [Qehk] 0    BMij
01001101101010100000101000011101111111  222 BEFHIKMOUWbcd          | KOW    222 [FUbc] 11   dfg    1    hk     1    BEHIMij
01001000001010100000011000011000111111  11  BEKMOVWbc              | KOW    11  bc    11   Vg     1    hk     1    BEMij
00010010001000101110111000110011111001  222 DGKOQRSUVWabe          | [DKOSW] 11 Ub    11   Vfg    1    [Qehk] 0    GRa
00011100001000100000010000011100101 11  0   DEFKOVbcd              | DKO    0   Fbc   0    Vdg    0    k      0    Eij
00010000101010110010011000011111111111  222 DIKMOPSVWbcde          | [DKOSW] 11 b     11   [Vdfg] 0    ehk    0    IMPij
00010100001000100000001000001001101 11  0   DFKOWc                 | DKOW   0   Fc    0    fg     0    k      0    ij
00010000101100101000010100001101101 11  11  DIKLOQVXcd             | DKO    11  c     11   [Vdfg] 0    Qk     0    ILXij
00010000001100101010101100000001111111  222 DKLOQSUWXe              | [DKOSW] 11 U     11   fg     1    [Qehk] 0    LXij
00010000101000100000011100000001110111  0   DIKOVWXe               | DKOW   0         0    Vfg    0    ek     0    IXij
00010000001100111010001000010101101 11  11  DKLOPQSWbd             | [DKOSW] 0  b     0    dfg    0    Qk     0    LPij
00011100101000100001100000001101111111  11  DEFIKOTUbce            | DKO    11  [FUbc] 0    fg     0    ehk    0    EITij
```

本来就已经凑好了 24 题，这个时候 ffzero 又想到一个很好的 idea（猜猜是哪题），如果直接将那道题加进来这个 Meta 就炸了！最后经过讨论只能把一个还没出好数据的比较简单的题目踢掉~~在客观上增加了这场愚人节比赛的难度~~，又调整了一题的题面和位置，这题的出题过程也就终于结束了。

最终本题的解法是：

首先，将所有 24 题对应的 Bingo 盘面做出来，并提取出这个 Bingo 连成了几条线，得到 `001010010012121202012011`。

根据第一行第四个描述的提示，得到这个三进制串后三位一断，再 A1Z26（“A001Z222”）提取为字母，就可以得到答案 `accepted`。

当然如果你已经看出来这题的提取方式了，那么不需要把所有题目的盘面都做出来，得到前几个字母应该就能把答案爆出来了。

附上所有题目的 Bingo 盘面（按题号顺序从左到右，从上到下）：

感叹：最后还是出不成真正的 Meta 啊，就算只看 PH 区，各个小题的答案也都太混乱了。

【CTF】

M / RuShiA

By ffzero

- Case 1：暴力分解攻击

理解了题意暴力分解做就行。

- Case 2：Pollard-Rho 分解攻击

观察发现 p 有 300 位左右，这说明 p 和 q 的差距很大，Pollard-Rho 算法擅长的就是这种分解。当然暴力应该也能很快分解出来。

- Case 3：费马分解攻击

一个比较新鲜（但在 OI 里几乎用不上）的分解算法：费马分解。费马分解适合两个质数相差很小的情况。

令 $a=\frac{p+q}{2}$，$b=\frac{p-q}{2}$，则有 $n=a^2-b^2$。枚举 a^2（即枚举比 n 大的完全平方数），发现 a^2-n 亦为平方数时即分解成功。本题 $p-q\leq 10^3$，很快就分解出来了。

- Case 4：最大公约数攻击

送分点，算最大公约数。

- Case 5：小指数明文爆破攻击

思路打开：$e=3$ 太小了，n 很大，有没有可能 $m^e\leq n$? 本 case 里直接对 m 开立方根就做完了。

当然不是所有情况都像本 case 这么极端，通用一点的做法是：根据关系式 $c=m^e \bmod n$，则必有一常数 k 使得 $kn\leq m^e\leq (k+1)n$。枚举 k，当发现 $kn+c$ 能开 e 次方根的时候即攻击成功。

- Case 6：共模攻击

比较带有 CTF 元素的一点是要让选手自己注意到 $n_1=n_2$。这点在特殊性质里没写。

发现这个性质之后就是送分点，用拓展 gcd 解方程组而已。

- Case 7：广播攻击

这里 $e=3$，但是用 Case 5 的办法做不出来。思考数学性质：

$$
\begin{aligned}
c_1 &\equiv m^e \bmod n_1\\
c_2 &\equiv m^e \bmod n_2\\
c_3 &\equiv m^e \bmod n_3
\end{aligned}
$$

使用中国剩余定理必能解出一个 c_x 使得

$$
c_x\equiv m^e \bmod \prod_{i=1}^{e} n_i
$$

然后对 c_x 开 e 次方根即可。本题给的方程组数量正好和 e 相同。

- Case 8：额外信息攻击

事实上比其他 case 都简单一些。已知 $n=pq$，$a=(p+2)(q+2)=pq+2(p+q)+4$，则可以得知 $p+q$ 的值。解二次方程即可。

- Case 9：明文泄露

（本来要放 LLL 的后来想想算了）

注意到已经给出 m。明文泄露了直接提交就得分了。

- Case 10：线性相关攻击 被删了

N / Web

By ffzero

代码审计题（读代码找 bug）。有些东西真的很难解释清楚，如果没有背景知识的话……

一句话题解：JSON 反序列化采用覆盖模式，而且有 GBK 宽字节抗转义注入。

发现一个小问题是在 JSON 反序列化的时候，如果一个键值出现了两次，后面的值会把前面的覆盖掉。所以，如果能够想办法在 JSON 字符串的末尾添加一个 `"admin":"true"`，那就能覆盖掉前面的 `false` 注册成为管理员。

直接拿 `"admin":"true"` 当用户名是不行的，需要闭合前面的引号。然而发现，程序会对输入的字符串里的双引号进行转义。但是又能发现，程序遇到 `0×80` 等特殊字节会跳过后面的字符不进行转义，如果在引号前面加入一个字节比如 `0×df`，就会让程序略过后面的双引号不转义。另外，JSON 反序列化的时候遇到右花括号 `}` 就会直接停止运行。所以注入完之后立刻添加个 `}` 字符就可以了。

这就是 GBK 编码注入抗转义，然后就会成功让双引号不被转义，然后闭合引号实现逃逸。接着还会发现，处理 JSON 的时候还是允许使用单引号的，所以可以用单引号的版本：`‘admin’:‘true’`。

最终答案是：`http://localhost:4080/register/%df','admin':'true'}`。

题外话：用 C++ 写的代码还是太裸了……如果用其他会自己处理 GBK 宽字节的语言（也就是，代码里不会出现一个那么明显的 `if` 语句）估计难度会大得多。

另外，原本那个 MD5 是用来测试的……没想到赛时那么多人直接爆破 MD5 过了……难受

O / RTX

By ffzero

运行程序，随便玩两下，或者拉进 IDA 之类的反编译软件，就会发现一个提示：`actually there is something behind the end of this program`。这里的 end 指的并不是程序控制流的末尾，而是程序文件的末尾。在末尾会发现一大段字母，只要有点经验就会发现这是 Base64 编码。

解码之后会得到一个二进制文件——如果对二进制敏感性不足的话，可以上 checkfiletype.com 让它识别文件格式（当然也可以 winhex 或者 binwalk 或者肉眼瞪文件头）。得到的是一个 zip 文件。

这个 zip 文件里面三个文件，`encoder` 显示为被加密的状态，有个空文件 `there_is_no_password.txt`，还有一个二进制文件 `enc.bin`。CTF 常用的技巧之一——伪加密的 zip 压缩包。原理不多概述，自行查阅。用十六进制编辑器把加密标志位改回来即可解压。当然选择直接 `-ff` 或者修复也行。

解压之后看 `encoder` 内容是源代码，从文件名等因素综合能判断出 `enc.bin` 是加密后的文件，解密需要密钥。那么密钥呢？硬着头皮读代码可以看到第 14 行（代码经过美化处理）：

```
text.push_back(key.size());
while(key.size()){
    text.push_back(key.back()*16+16-key.back()/16&0xff);
```

```
    key.pop_back();
}
```

这个加密程序会做如下事情：先把密钥的长度放到文件末尾，然后倒序把密钥的每个字节进行运算 `b * 16 + (16 - b / 16)` 之后放到末尾。这个运算也很简单：只是把一个字节的高四位取反之后和低四位互换，比如 `0x41`，变为 `0xC1` 然后再变为 `0x1C`。

再研究上面的算法，会发现对于相同的密钥，算法会生成同一个序列值，把明文的每一位依次异或，所以加密和解密用的是同一套流程——把密文用密钥再次加密就会得到明文。当然如果是 CTF 选手来做，就很容易发现这个是著名的 RC4 加密算法。

综上，可以逆向出密钥然后解密文件。有了之前的经验，就会发现这又是个 zip 压缩包。解压后得到一个文件，下面一大片 `!+()[]` 组成的字符一看就知道是 JS 加密（拿去问 Deepseek 或者直接搜索都能知道）。粘贴到浏览器里运行会得到一堆字符。

这堆字符看上去像是英文但是没有一个是认识的词——单表替换法。人肉解单表替换的方式不多讲了，这里给出一个方便的网站 quipqiup.com，专门破解这类密码。总之，破译出来是一行提示，让你取这条消息所用的替换字母表的前十二个字母组成答案——`miskchalengr`。用 `flag{}` 包裹后提交即可通过。

题外话：对 OI 选手来说可能很漫长而且每一步都不好想，这题很难——其实并非如此，主打一个认知差。本题其实很多地方对选手够仁慈了。比如 `encoder` 完全可以放编译后的可执行程序，这就硬性要求选手有反编译能力了。

如果身边有玩 CTF 的同学的话，试试看让 ta 来做吧，真的不难。

【FUN】

P / 数学题

By cff_0102

题面多次提示你有英文题面，正好最近洛谷上线了多语言题面，本题就可以在最上面切换语言为英文。

切换为英文后再看最后的提示，就告诉了你实际上题目的编号是按照首字母顺序排的。

如果没发现这个，也可以交最多 10 发来确定每个题目对应的编号。

接下来的题解是按题目编号的顺序排列的。

A. 你可以直接算，也可以把这段描述搜一下，就可以搜到 OEIS 的 A051049。虽然那个序列的长度太短没有告诉你这题的正确答案，但是它告诉了你怎么推。手动计算就可以得到答案是 2147483647。

C. 这里的字体和控制台的相符，在控制台计算 $0.1+0.2$ 的结果是 0.30000000000000004。

F. 找规律，也是在 OEIS 搜一下就可以了，注意力好一点的可以直接看出这个序列是什么。OEIS 编号是 A000127，下一项是 99。

G. 题目截取自 P1618 三连击（升级版）。给这题正解代码输入 `1 3 5`，就可以得到答案为 `129 387 645`。

O. 我们都知道 $e^{i\pi}=-1$，所以小数点位后面全是 0。输出 $1919810-114514+1$ 个 `0` 即可。

P. 随便找个进制转换器，就可以得到答案是 6737151。

S. 随便解即可，答案是 19。中英文版的图都是从 WolframAlpha 上面截的。

T. 截取自 CSP-J 2020 初赛的第四题，答案是 C。

W. 写个 Python 代码跑一下，可以得到答案是 450。

Y. 虽然图被挡住了一部分，有些点权看不到，但是边权没被挡住，只需要计算最小生成树的边权之和即可。答案是 11。

AC Code:

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
  int x;cin>>x;
  switch(x){
    case 1:
      cout<<"2147483647";break; // https://oeis.org/A051049
    case 2:
      cout<<"0.30000000000000004";break; // 0.1 + 0.2
    case 3:
      cout<<"99";break; // https://oeis.org/A000127
    case 4:
      cout<<"129 387 645";break; // https://www.luogu.com.cn/problem/P1618
    case 5:
      for(int i=114514;i<=1919810;i++)cout<<0;break; // e^(i*pi) = -1
    case 6:
      cout<<0x66ccff;break; // 0x -> hex
    case 7:
```

```
      cout<<"19";break; // https://www.wolframalpha.com/（太长省略）
    case 8:
      cout<<"C";break; // https://ti.luogu.com.cn/problemset/1034 T4
    case 9:
      cout<<"450";break; // You may use Python
    case 10:
      cout<<"11";break; // ez
  }
  return 0;
}
```

Q / Time goes back

By lndjy

这题是 2024 年出的，经过讨论出题组认为放在 2025 更好，于是就今年放了。

有三条线索指向答案在去年比赛的 A 题 Nine Gates：

- 时间倒流提示去以前的题目中寻找答案。
- 九是 nine，大门是 gate，这指向了去年比赛中 Nine Gates 这道题。
- 两题题目描述中都有小 D。

门出现的时间，自然就是 Nine Gates 的出题时间，在 Nine Gates 底部的剪贴板，有发布时间：2024-03-08 8:27，在这个时间之前的中午 12 点就是 2024-03-07，也就是答案。

R / 打字练习

By ffzero

有很多解法。打到准度超过 95% 就通关。

- 叠力神：把 Legend 打过。
- 科技哥：写脚本等手段模拟按键（不知道为什么 `keybd_event` 之类的不行，无所谓了）。
- 梅西爷：反编译程序。这个题用的 `pyinstaller` 打包。用 `pyinstxtractor` 就能反编译出代码了，然后直接提交。这也是可以接受的解。
- 预期正解：

翻一下游戏文件夹的 `s` 目录，里面存储的是游戏美术资源（比如图标和音频），发现里面有两个音频文件。结合 `Copyright.txt`，这首歌叫《幸せになれる隠しコマンドがあるらしい》，意思是“有着能够变得幸福的隐藏指令”（音游圈内有简称《幸隐》）。

- 题面为何如此频繁地强调“隐藏”，还说有“隐藏”关卡？
- 在选关卡的时候，窗口标题栏居然在显示按下的方向键（↑↓←→）？
- 什么叫用方向键选择关卡？隐藏关也可以用方向键吗？
- 搜过这首歌了吗？歌词开头也是方向键？

总之，如果在选难度的时候按下《幸隐》开头的方向序列（自行查阅），就能进入隐藏关了。

不过隐藏关不给通关令牌，取而代之的是提示 `slow down and look`，而题面里也提示了“请慢慢来”。解法就是真正的慢慢来，配置文件 `config.txt` 里把流速调到很低（比如 0.1）就会发现，隐藏关的音符的排列，那些空隙形成了数字……正好十一个数字，这就是隐藏的数字指令了。提交即可拿下满分。

简单科普一下：Legend 难度是 4k（四轨定轨下落）圈内 Extra Final 段位关。这个谱是人能打的，但是打过这个关（通关条件和本题不一样）的都可以算是 4k 里的 top 玩家了（

原本想上 1.2 大天空或者 sendan 的（

S / LUA

By ffzero

标题 LUA 但是代码用 Python？其实标题的意思是鲁 A。

直接跑 Python 的话会断言失败，因为答案长度不是 15。

看到一个奇怪的字符串 `origin`，其实意思是“源”，视频源的“源”，后面的数字字母串其实看着就很像一个 B 站视频号。此时就不难想到：`origin[0:pos]` 可能指的不是 Python 里的切片（slice）操作，而是指原视频的 0 分 `pos` 秒。

视频是“鲁 A 济南车 鲁 B 青岛的”，如果把对应秒数的字母连接起来，就会得到 `KFCVME`，再和后面的 Python 程序连接起来，答案是 `KFCVME50_xpkman`，提交即可。

赛时评论区有人直接爆答案，也是这种视频题的缺点吧。

T / 奇怪的画面

By E.Space

题目背景中的文字暗示这个游戏是扫雷。

提交任意内容之后会发现有 49 个测试点且颜色分布比较混乱，对应了题目背景中“五颜六色的画面”的描述。

注意洛谷的评测结果页面一行最多会显示 7 个测试点，但是当放大页面后或屏幕宽度较小时，每行显示的测试点数量可能会小于 7；但是反过来，如果缩小页面或者屏幕宽度较宽时，每行仍然最多只会显示 7 个测试点。

输出格式要求输出若干行 01 串，可以猜测需要输出每个位置是否是雷。

关于评测结果的含义的提示仍然是颜色。注意到经典扫雷画面中数字 1 是蓝色的，数字 2 是绿色的，数字 3 是红色的，数字 4 是深蓝色的，这正和评测结果中 `UKE`，`AC`，`WA`，`TLE` 的颜色类似。

每个数字表示周围八个格子中雷的数量。虽然有可能第一反应是九个，即包含格子自身，但是通常情况下可以发现这是无解的。

在每个整点，本题的地图会发生变化。解出雷的位置后提交即可通过。你可以用 `1` 表示雷，`0` 表示空地；也可以用 `0` 表示雷，`1` 表示空地，这两种提交都可以通过。甚至，如果真的发生了小概率情况，也就是当数字表示周围九个格子的雷的数量时也有解时，使用这个理解来输出答案也可以通过。

U / 看谱识曲

By Eason_cyx

Fun Fact：这个题是 Eason 向本次比赛中供的 7 个题里唯一一个选上的（

首先根据样例 & 提示去听 Rrhar'il，发现箭头和旋律完全对不上，所以提示就是误导你的。

观察题目中的箭头：↖，↙，↗，↘，→，←。发现他们很可疑——一般这种箭头都不会用←，↙和↖！于是就想到，其实这里只是把歌名看作一个字符串，那么进一步，就可以发现这实际上是在键盘上的移动操作，空格表示移动到了这个字母。手玩一下样例发现这是对的，是从 Q 开始移动的。这也就解释了为什么歌曲名被去掉了标点和空格。

另外，输入格式中使用的字母也提示了箭头和键盘有关。

那么模拟坐标移动即可，记输入字符串为 S，则时间复杂度是 $O(|S|)$。

std：

```cpp
#include <bits/stdc++.h>
using namespace std;
char keyboard[45][205] = {  "                    ",
                            " q w e r t y u i o p",
                            "  a s d f g h j k l ",
                            "   z x c v b n m    "};
int main() {
  int sx = 1, sy = 1; string s, ans; while(cin >> s) {
    for(int i = 0;i < s.size();i++) {
      if(s[i] == 'W') sx--, sy--;
      if(s[i] == 'E') sx--, sy++;
      if(s[i] == 'Z') sx++, sy--;
      if(s[i] == 'X') sx++, sy++;
      if(s[i] == 'A') sy -= 2;
      if(s[i] == 'D') sy += 2;
    } ans += keyboard[sx][sy];
  } cout << ans << endl;
  return 0;
}
```

Fun Fact 2: 本题的数据都是真实的音游曲目。包含了 Phigros 和 Dancing Line 中的一些曲子，如有需要可以找 Eason 要（?

V / 别样的程序 · 改

By ffzero

从题目背景和标题里看都能看出这个题参考了未来程序·改，很著名的大模拟题，要让人写个 C++ 编译器的那种。

要提供 std? 没错，是要提供，但是提供指的是要输出源代码啊！当然也不是什么源代码都能过的，提示里写的“请注意使用正确的 C++ 语言”意思是要使用 未来程序·改 里定义的 C++ 语言。

这题的 SPJ 会把你的输出当做 C++ 代码直接执行。没错，够暴力……但是这题确实是这样。

回到要解决的题目本身，题意简述：在 Minecraft 中，挖了一个 $n \times m$ 的坑，往里面倒几桶水能完全灌满。如果一个格子上下左右有至少两个格子是水源，那么它也会变成水源。

题解：先尽可能对角放，然后把剩下的部分沿着一条边隔一格放一格。答案是 $\frac{n+m+1}{2}$ 下取整。证明不难。

其实可以写完 C++ 然后用 PHP 交，因为 PHP 会把代码（在没有标签的情况下）原样输出。

W / CTF

By ffzero

定位是冲提交量的题，需要多次提交来获取提示。但是好像意识到这题在干什么的选手比较少，导致最后提交量低于预期了。

赛时看了一堆人用各种文件读取方式读 `flag.txt`。没错是在评测机里，但是死活就是读不到。

如果尝试输出点东西就会发现评测机返回的东西似乎有点意思：末尾都是黑的（TLE）开头都是红的（WA），而且评测时间和评测空间都是一样的整数！是在强调什么?

如果对数字敏感或者玩 CTF 几乎一眼就能看出这是 ASCII 编码。然后翻译一下就会发现真的有东西。结合题目背景里所说的“Linux 大佬”，容易看出这返回的是 Linux 命令行输出。建议写一份代码用来翻译评测结果。

`flag.txt` 确实在评测机上，但是在评测机上运行的一个虚拟机里。SPJ 是一个模拟 Linux 存储文件的虚拟系统（这个模拟版 Linux 是人写的，有些指令以及某些指令参数是不支持的）。

其实这个题模拟的是网页挂了 Webshell（当然也可能是挂了一句话木马）。

先说标准答案，一行指令即可通过：`curl file:///var/F/3/5/B/flag.txt -o /tmp/answer.txt`。

首先先 `ls` 发现可以运行，接着就会发现一个 `network-test.txt`，里面有些 `curl` 指令。然后 `pwd` 一下发现在 `/home/guest/` 里。然后 `cd ..` 到父目录，再 `ls` 一下能发现点东西：一堆用户。有 `admin` 文件夹但是里面似乎没什么有用的东西。

“有些东西是看不见的”，这提示需要 `ls -a` 查看隐藏的文件和目录。然后就可以看到多了一个 `.anti-attack` 文件夹，里面有一堆日志文件。这里不具体列出所有文件了，总之可以获取一些有用的东西：有文件被移动到了 `/var/4/0/8/0/`；flag 会十秒改变一次；评测机会通过比对 `/tmp/answer.txt` 中的内容来判断是否正确。

先去 `/var/4/0/8/0/`，发现有一个 `declaration.txt`，里面有神秘字符串 `f35b`，接着再去 `/var/F/3/5/B/` 就会发现 `flag.txt` 了……所以这题做完了吗？

没做完，`cat flag.txt` 会被 anti-attack 拦下来。试图绕过？`tac`、`rev`、`head`、`tail`、`more`、`less`、统统都不行（你能想到的出题人都能想到）。

其实如果想到了读取 `flag.txt` 的途径这题也就差不多做完了。这道题明明只提供了基础的命令，为什么还会有 `curl`？可以剑走偏锋：尝试用 `curl` 读取文件。

要知道 `http://` 或者 `https://` 什么的都只是协议好吗……有个本地文件系统的协议 `file://` 也可以用……老司机可能用过 `magnet://`（对）。

输出到控制台的参数是 `-O`，输出到文件的参数是 `-o`。这道题最好用文件输出（也更方便），因为 flag 十秒变一次，用 `-O` 输出然后解码再提交的方式可能会来不及（验题的时候拼手速成功过，不过注意要 `echo -n`，不然会多换行符）。

闲话：

本题是本场里首 A 最晚出现的题目（比赛刚开始的时候有 bug 导致 RE 的程序可以直接通过此题，但是数据修复后重测了，这不算首 A）。难度可能确实有点超出预期，有很多选手都卡在第一步的翻译 ASCII，不知道需要输出 Linux 命令，或者不知道 `cd ..` 跳出去有更多东西而一直卡在刚开始的地方。

起初，本题的 flag 是固定的，且不需要输出到特定文件，直接当作“命令”输出。赛前发现这样翻提交记录找答案的概率很大，验题人 (cff) 想到只要在某个隐藏的地方引导选手输出到特定位置，翻提交记录的选手就不知道应该输出到哪了。而且正解输出全部都是命令，也更自然一点。

改完之后，还是担心有人不知道 `curl` 读文件而通过翻提交记录跳过关键的一步，邪恶的出题人就给 flag 加上了十秒变一次的限制。所以 `.access_logs` 中加上这个限制的时间 `2025-03-30` 是真实的。

出题人们的节目之一：翻提交记录看选手用各种花样操作读取 `flag.txt`。包括但不限于：

- 试图删除 anti-attack；
- 试图 `mv` 或者 `cp` 那个 `flag.txt` 到其他地方然后读取的（被 anti-attack 拦了）；
- 试图 `vim` 之类的（出题人写代码写累了真的懒得实现一切命令）；
- 试图删掉整个系统 `rm -rf /`（删了也没用）；
- 试图把 `flag.txt` 和 `answer.txt` 用 `echo` 覆写成一样的（出题人没想到有这种操作，但是没关系因为也过不了题）；

同时，本题存在一个隐藏奖励：

在 I 题（《拼好题》）的第四道“子问题”中，有一句话不存在于组成它的任何一道题目：“已知 $x! \leq 2^k$，你需要找到 k 对应的路径。”

正好，P 题（《数学题 / Math Problems》）的问题 9（首字母为 W）同样出现了“$x! \leq 2^k$”，其中 $k = 3327$。

注意到 3327 有四位，本题 `var` 中的“迷宫”也有四层，猜测在 `/var/3/3/2/7/` 中有东西。可以发现其它错误路径里面的文件都是叫 `.keep` 的隐藏文件，而这里的隐藏文件名变成了 `.egg`。读取里面的内容（这个可以直接用 `cat`），可以得到 `send EhhAcroSsThrEepRoblems to SKY's Luogu account`（刚好 50 字符）。因此要把 `EhhAcroSsThrEepRoblems` 发送到一个洛谷用户的私信中。

由 P 题的中英文对照可知小 S 的英文名是 Sky，小 A 的英文名是 Aqua，正好出题人列表中 cff_0102 的 tag 就是 `sky & aqua`，因此直接将上述内容私信给 cff 即可。赛后就不要骚扰了。

至于为什么这里的 `egg` 会拼错，起因是本题出题人在赛时修 bug 的时候直接用的本地的数据文件而不是重新下载一遍数据，而出题人本地的数据没有加这个彩蛋。后来被 cff 发现，焦急的催促出题人把彩蛋加回去，这个时候就把单词拼错了。后来发现也懒得改了，不失为一种防伪标识。

X / Special Thanks 3

By cff_0102

与前年的 Special Thanks 2 一样，按顺序输出这场比赛每题的出题人即可……吗?

由于题目要求输出 13 行，而本场比赛共有 24 题，说明输出的不是今年愚人节比赛的题目。

注意到去年的愚人节比赛没有 Special Thanks，且去年恰好有 13 题，因此本题需要输出的是去年的出题人列表。可以在去年愚人节比赛的题解中看到。

本题没有部分分，只有全部答对才能得分。