囚徒挑战（prison）

一个监狱里关着 500 名囚徒。 有一天，监狱长给了他们一个重获自由的机会。 他把装钱的两个袋子 A 和 B 放在一个房间里。 每个袋子装有若干枚硬币，数量的范围在 1 到 N 之间（包含 1 和 N）。 两个袋子里硬币的数量不同。 监狱长给囚徒们提出了挑战，目标是指出硬币数量较少的那个袋子。

房间里除了袋子还有一块白板。 任意时刻白板上写着一个数，一开始写的是 0。

监狱长让囚徒一个接一个地进入房间。 每个进入房间的囚徒不知道他之前进入过房间的囚徒有多少人，也不知道是哪些人。 每次一个囚徒进入房间时，他看一眼白板上目前写的这个数。 看完之后，他必须在袋子 A 和 B 之间做出选择。 接着，他检查自己选的那个袋子，知道了里面有多少枚硬币。 然后，这名囚徒必须选择做以下两种行动之一：

- 将白板上的数改写成一个非负整数，并离开房间。 注意他可以改变成新的数，也可以保留当前的数。 然后挑战继续进行（除非所有 500 名囚徒都已经进过房间）。
- 指出硬币数量较少的那个袋子。这会立即结束挑战。

对于已经进过房间的囚徒，监狱长不会让他再次进入房间。

如果某个囚徒正确地指出硬币较少的袋子，则囚徒们获得挑战的胜利。 如果指出的袋子不正确，或者所有 500 人进过房间之后还没有人尝试指出硬币较少的袋子，则囚徒们失败。

挑战开始之前，囚徒们集合在监狱大厅商量应对挑战的共同策略，分以下三个步骤：

- 他们挑选一个非负整数 x，作为他们可能会写在白板上的最大的数。
- 他们决定对任意一个数 i $(0 \leq i \leq x)$，如果某个囚徒进入房间后看到白板上写着数 i，那么他应该去检查哪个袋子。
- 他们决定当某个囚徒得知选中的袋子里的硬币数量后要采取的行动。具体来说，对任意写在白板上的数 i $(0 \leq i \leq x)$ 和检查选中的袋子里的硬币数量 j $(1 \leq j \leq N)$，他们要决定做出以下两种行动之一：
  - 白板上应该要写一个 0 到 x 之间（包含 0 和 x）的什么数；
  - 指出哪个袋子是硬币较少的。

如果赢得挑战，监狱长会在囚徒们继续服刑 x 天后释放他们。

你的任务是提出能够确保囚徒们赢得挑战的策略（不管袋子 A 和 B 中的硬币数量是多少）。 你的得分取决于 x 的值（详见子任务一节）。

实现细节

你要实现以下函数：

```
int[][] devise_strategy(int N)
```

- N：每个袋子里硬币最多可能的数量。
- 该函数需要返回一个数组 s，它的每个元素是长度为 $N+1$ 的整数数组，表示你给出的策略。 x 的值是数组 s 的长度减一。 对满足 $0 \le i \le x$ 的每个 i，数组 $s[i]$ 表示囚徒在进入房间看到白板上写着数 i 时要做的事情：
    1. 如果囚徒应该检查袋子 A，则 $s[i][0]$ 的值是 0；如果囚徒应该检查袋子 B，则该值是 1。
    2. 令 j 为所选袋子中的硬币数量，囚徒应该进行以下行动：
        - 如果 $s[i][j]$ 的值是 -1，则囚徒应该指出袋子 A 是硬币较少的袋子。
        - 如果 $s[i][j]$ 的值是 -2，则囚徒应该指出袋子 B 是硬币较少的袋子。
        - 如果 $s[i][j]$ 的值是非负整数，则囚徒应该把这个数写到白板上。注意 $s[i][j]$ 至多只能是 x。
- 该函数恰好被调用一次。

例子

考虑以下调用：

```
devise_strategy(3)
```

令 v 表示囚徒进入房间时看到白板上写着的数。以下是一种正确的策略：

- 如果 $v = 0$（也包括开始时的数），则检查袋子 A。
    - 如果它装了 1 个硬币，则指出袋子 A 是硬币较少的袋子。
    - 如果它装了 3 个硬币，则指出袋子 B 是硬币较少的袋子。
    - 如果它装了 2 个硬币，则在白板上写上 1（覆盖之前的 0）。
- 如果 $v = 1$，则检查袋子 B。
    - 如果它装了 1 个硬币，则指出袋子 B 是硬币较少的袋子。
    - 如果它装了 3 个硬币，则指出袋子 A 是硬币较少的袋子。
    - 如果它装了 2 个硬币，则在白板上写上 0（覆盖之前的 1）。注意，这种情况其实不可能发生，因为此时两个袋子都装有 2 枚硬币，是不允许的。

要产生以上策略，函数应该返回 `[[0, -1, 1, -2], [1, -2, 0, -1]]`。 返回的数组长度是 2，此时 x 的值是 $2 - 1 = 1$。

约束条件

- $2 \le N \le 5000$

子任务

1. （5 分）$N \le 500$，x 的值不能超过 500。

2. （5 分）$N \leq 500$，x 的值不能超过 70。
3. （90分）x 的值不能超过 60。

对于任何测试用例，如果 `devise_strategy` 返回的数组是不合法的，则你在该子任务上的得分为 0。

子任务 3 有部分分。 令 m 为该子任务中所有测试用例返回数组对应的 x 的最大值，你的得分将根据下表计算：

条件	得分
$40 \leq m \leq 60$	20
$26 \leq m \leq 39$	$25 + 1.5 \times (40 - m)$
$m = 25$	50
$m = 24$	55
$m = 23$	62
$m = 22$	70
$m = 21$	80
$m \leq 20$	90

评测程序示例

评测程序示例按以下格式读取输入：

- 第 1 行：N
- 第 $2 + k$ 行 $(0 \leq k)$：$A[k]$ $B[k]$
- 最后一行：-1

除第一行和最后一行外，每行表示一个场景。 将第 $2 + k$ 行对应的场景称为场景 k。 场景 k 中，袋子 A 装有 $A[k]$ 枚硬币，袋子 B 装有 $B[k]$ 枚硬币。

评测程序示例首先调用 `devise_strategy(N)`。 x 的值是返回数组的长度减一。 如果评测程序示例检测到 `devise_strategy` 返回的数组不符合实现细节中描述的约束，它会打印如下错误信息并退出：

- `s is an empty array`：s 是空的数组（表示不合法的策略）。
- `s[i] contains incorrect length`：存在一个下标 i $(0 \leq i \leq x)$ 满足 $s[i]$ 的长度不是 $N + 1$。
- `First element of s[i] is non-binary`：存在一个下标 i $(0 \leq i \leq x)$ 满足 $s[i][0]$ 既不是 0，也不是1。
- `s[i][j] contains incorrect value`：存在下标 i, j $(0 \leq i \leq x, 1 \leq j \leq N)$ 满足 $s[i][j]$ 的值不在 -2 和 x 之间。

否则，评测程序示例产生两项输出内容。

首先，评测程序示例以如下格式打印你的策略的输出：

- 第 $1+k$ 行 ($0 \le k$)：场景 k 下你的策略的输出。 如果用该策略导致某个囚徒指出袋子 A 是硬币较少的，则输出字符 `A`。 如果用该策略导致某个囚徒指出袋子 B 是硬币较少的，则输出字符 `B`。 如果用该策略后没有囚徒指出哪个袋子的硬币较少，则输出字符 `X`。

其次，评测程序示例以如下格式在当前目录下写一个文件 `log.txt`：

- 第 $1+k$ 行 ($0 \le k$)：$w[k][0]$ $w[k][1]$ ...

第 $1+k$ 行的序列对应于场景 k，描述了写在白板上的数。 具体来说，$w[k][l]$ 是第 $l+1$ 个囚徒进入房间后写的数。