Fängelseutmaning

I ett fängelse finns det 500 fångar. En dag erbjuder vakten dem en chans att bli fria. Han lägger två kassar med mynt, kasse A och kasse B, i ett rum. Varje kasse innehåller mellan 1 och N mynt, inklusive. De två kassarna innehåller **olika** antal mynt. Vakten ger fångarna en utmaning. Fångarnas mål är att identifiera vilken kasse som har minst antal mynt.

Förutom myntkassarna innehåller rummet även en anslagstavla. Anslagstavlan måste alltid visa exakt ett heltal. Till att börja med är detta heltal 0.

Därefter ber vakten fångarna att gå in i rummet en efter en. Fången som går in i rummet vet inte vilka eller eller hur många andra fångar som har gått in före. Varje gång en fånge intar rummet så läser den heltalet som tavlan då visar. Sedan väljer fången antingen kasse A eller kasse B. Fången **undersöker** den utvalda kassen, och får då veta antalet mynt inuti. Därefter utför fången en av följande **handlingar**.

- Skriv över heltalet som tavlan visar med ett icke-negativt heltal, och lämna rummet. Notera att de antingen kan ändra eller behålla det nuvarande heltalet. Utmaningen fortsätter sedan (förutom om alla 500 fångar redan har varit i rummet).
- Identifiera en kasse som den med minst antal mynt. Detta avslutar omedelbart utmaningen.

Vakten kommer inte be en fånge som har lämnat rummet att gå in igen.

Fångarna vinner utmaningen om en av dem korrekt identifierar kassen med minst antal mynt. De förlorar om någon av dem felidentifierar kassen, eller om alla 500 fångarna har gått in i rummet och ingen har försökt hitta kassen med minst antal mynt.

Innan utmaningen börjar har fångarna samlats i matsalen och bestämt en gemensam **strategi** för utmaningen, i tre steg.

- De väljer ett icke-negativt heltal x, som är det största heltalet de kommer vilja skriva så att anslagstavlan visar det.
- De bestämmer, för varje heltal i som tavlan kan visa ($0 \leq i \leq x$), vilken kasse som ska undersökas av en fånge som läser heltalet i från tavlan när denne äntrar rummet.
- De bestämmer vilken handling en fånge i rummet ska utföra efter att ha sett hur många mynt som finns i en viss kasse. Mer exakt, för varje tal i som tavlan kan visa ($0 \leq i \leq x$) och varje antal mynt j som kan synas i kassen ($1 \leq j \leq N$), så bestämmer de antingen
  - vilket tal mellan 0 och x (inklusive) som ska skrivas så att tavlan visar det, eller
  - vilken kasse som ska identifieras som den med minst mynt.

När fångarna vinner utmaningen, kommer vakten låta fångarna gå efter x ytterligare dagar.

Du ska finna en strategi som fångarna kan använda för att säkerställa att de vinner utmaningen (oavsett hur många mynt som finns i kasse A och kasse B). Din kod kommer att bedömas utifrån värdet av x (se Subtasks nedan).

Implementation Details

You should implement the following procedure:

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

- N: the maximum possible number of coins in each bag.
- This procedure should return an array s of arrays of $N + 1$ integers, representing your strategy. The value of x is the length of array s minus one. For each i such that $0 \leq i \leq x$, the array $s[i]$ represents what a prisoner should do if they read number i from the whiteboard upon entering the room:
    1. The value of $s[i][0]$ is 0 if the prisoner should inspect bag A, or 1 if the prisoner should inspect bag B.
    2. Let j be the number of coins seen in the chosen bag. The prisoner should then perform the following action:
        - If the value of $s[i][j]$ is -1, the prisoner should identify bag A as the one with fewer coins.
        - If the value of $s[i][j]$ is -2, the prisoner should identify bag B as the one with fewer coins.
        - If the value of $s[i][j]$ is a nonnegative number, the prisoner should write that number on the whiteboard. Note that $s[i][j]$ must be at most x.
- This procedure is called exactly once.

Example

Consider the following call:

```
devise_strategy(3)
```

Let v denote the number the prisoner reads from the whiteboard upon entering the room. One of the correct strategies is as follows:

- If $v = 0$ (including the initial number), inspect bag A.
    - If it contains 1 coin, identify bag A as the one with fewer coins.
    - If it contains 3 coins, identify bag B as the one with fewer coins.
    - If it contains 2 coins, write 1 on the whiteboard (overwriting 0).
- If $v = 1$, inspect bag B.

- If it contains 1 coin, identify bag B as the one with fewer coins.
- If it contains 3 coins, identify bag A as the one with fewer coins.
- If it contains 2 coins, write 0 on the whiteboard (overwriting 1). Note that this case can never happen, as we can conclude that both bags contain 2 coins, which is not allowed.

To report this strategy the procedure should return `[[0, -1, 1, -2], [1, -2, 0, -1]]`. The length of the returned array is 2, so for this return value the value of x is $2 - 1 = 1$.

Constraints

- $2 \leq N \leq 5000$

Subtasks

1. (5 points) $N \leq 500$, the value of x must not be more than 500.
2. (5 points) $N \leq 500$, the value of x must not be more than 70.
3. (90 points) The value of x must not be more than 60.

If in any of the test cases, the array returned by `devise_strategy` does not represent a correct strategy, the score of your solution for that subtask will be 0.

In subtask 3 you can obtain a partial score. Let m be the maximum value of x for the returned arrays over all test cases in this subtask. Your score for this subtask is calculated according to the following table:

Condition	Points
$40 \leq m \leq 60$	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

Sample Grader

The sample grader reads the input in the following format:

- line 1: N

- line $2 + k$ ($0 \leq k$): $A[k]$ $B[k]$
- last line: -1

Each line except the first and the last one represents a scenario. We refer to the scenario described in line $2 + k$ as scenario k. In scenario k bag A contains $A[k]$ coins and bag B contains $B[k]$ coins.

The sample grader first calls `devise_strategy(N)`. The value of x is the length of the array returned by the call minus one. Then, if the sample grader detects that the array returned by `devise_strategy` does not conform to the constraints described in Implementation Details, it prints one of the following error messages and exits:

- `s is an empty array`: s is an empty array (which does not represent a valid strategy).
- `s[i] contains incorrect length`: There exists an index i ($0 \leq i \leq x$) such that the length of $s[i]$ is not $N + 1$.
- `First element of s[i] is non-binary`: There exists an index i ($0 \leq i \leq x$) such that $s[i][0]$ is neither 0 nor 1.
- `s[i][j] contains incorrect value`: There exist indices i, j ($0 \leq i \leq x, 1 \leq j \leq N$) such that $s[i][j]$ is not between -2 and x.

Otherwise, the sample grader produces two outputs.

First, the sample grader prints the output of your strategy in the following format:

- line $1 + k$ ($0 \leq k$): output of your strategy for scenario k. If applying the strategy leads to a prisoner identifying bag A as the one with fewer coins, then the output is the character A. If applying the strategy leads to a prisoner identifying bag B as the one with fewer coins, then the output is the character B. If applying the strategy does not lead to any prisoner identifying a bag with fewer coins, then the output is X.

Second, the sample grader writes a file `log.txt` in the current directory in the following format:

- line $1 + k$ ($0 \leq k$): $w[k][0]$ $w[k][1]$ $\ldots$

The sequence on line $1 + k$ corresponds to scenario k and describes the numbers written on the whiteboard. Specifically, $w[k][l]$ is the number written by the $(l + 1)^{th}$ prisoner to enter the room.