Multi Communication 2

President K has prepared a game for the contestants of the JOI Final Stage. President K secretly has an $N \times N$ table A, each of whose cells contains a non-negative integer. Let $A_{i,j}$ denote the integer written in the cell in the $(i+1)$-st row from the top $(0 \leq i \leq N-1)$ and the $(j+1)$-st column from the left $(0 \leq j \leq N-1)$. The table A satisfies the following conditions.

- For each i such that $0 \leq i \leq N-1$, we have $A_{i,i} = 0$.
- For each i, j such that $0 \leq i < j \leq N-1$, we have $A_{i,j} = A_{j,i}$.

Consider the weighted undirected graph with N vertices in which the edge between vertex i and vertex j $(0 \leq i < j \leq N-1)$ has weight $A_{i,j}$. Let X be the weight of a minimum spanning tree of this graph. In other words, X is the value obtained by the following procedure. The goal of the game is for all contestants to cooperate and determine the value of X.

1. Set $x = 0$.
2. Consider an undirected graph G with N vertices. The vertices of G are numbered $0, 1, \ldots, N-1$. Initially, G has no edges.
3. Repeat the following operation $N-1$ times. Let X be the value of x after these $N-1$ operations.
   i. Call the vertices of G that are currently reachable from vertex 0 by using some edges the **near vertices**, and call the other vertices the **far vertices**. Choose a pair (i, j) consisting of a near vertex i and a far vertex j so that the value $A_{i,j} \times N^2 + i \times N + j$ is minimized. It can be proved that there is at least one near vertex and at least one far vertex, and that (i, j) is uniquely determined.
   ii. Add the edge connecting vertex i and vertex j to G.
   iii. Update $x \leftarrow x + A_{i,j}$.

Define $R = \lfloor 5120/N \rfloor$ (where $\lfloor x \rfloor$ denotes the greatest integer not exceeding x). There are $R \times N$ contestants in the JOI Final Stage, divided into R groups of N contestants each. The groups are numbered 0 through $R-1$, and the contestants in group r $(0 \leq r \leq R-1)$ are numbered $(r, 0), (r, 1), \ldots, (r, N-1)$.

The game proceeds by repeating the following **rounds**, in the order round $0, 1, 2, \ldots$, at most R times. During the game, the contestants are not allowed to communicate with one another, but they may share a strategy in advance.

Round r $(0 \leq r \leq R-1)$ proceeds as follows.

- For $i = 0, 1, \ldots, N-1$ in this order, President K and contestant (r, i) perform the following interaction.
  1. President K gives contestant (r, i) the following information.
     - the contestant's number (r, i),
     - the information in the $(i+1)$-st row of the table A, namely $A_{i,0}, A_{i,1}, \ldots, A_{i,N-1}$,
     - the integers $B_{r,i,0}, B_{r,i,1}, \ldots, B_{r,i,N-1}$, each of which is an integer between 0 and $2^{64}-1$, inclusive, sent to contestant (r, i) from the contestants in the previous round.
       * If $r > 0$, these integers are determined in interaction 3 described below.
       * If $r = 0$, then for convenience we define $B_{r,i,0} = B_{r,i,1} = \cdots = B_{r,i,N-1} = 0$.
  2. If contestant (r, i) is able to determine the value of X, they answer President K with that value. If any contestant gives an answer, the game ends immediately.
  3. For each $j = 0, 1, \ldots, N-1$, contestant (r, i) determines an integer $B_{r+1,j,i}$, which must be between 0 and $2^{64}-1$, inclusive, to send to contestant $(r+1, j)$, and tells it to President K. Even when $r = R-1$, so that contestant $(r+1, j)$ does not exist, contestant (r, i) must still determine $B_{r+1,j,i}$ and tell it to President K.

If an answered value of X is incorrect, or if nobody answers the value of X by the end of round $R-1$, the game fails. If the correct value of X is answered by the end of round $R-1$, the game succeeds. A smaller number of rounds used yields a higher score (if an answer is given in round r, the number of rounds is counted as $r+1$).

Implement a strategy for the contestants so that the game succeeds in as few rounds as possible.

Implementation Details

Your submission must include `multi.h` using a `#include` preprocessing directive, and must implement the following function.

- `std::vector<unsigned long long> strategy(int N, int r, int i, std::vector<unsigned long long> A, std::vector<unsigned long long> B)`
  - The argument `N` represents the number of rows and columns of the table A.
  - The arguments `r` and `i` represent the contestant number (r, i).
  - The argument `A` is a sequence of non-negative integers of length N, where `A[j]` $(0 \leq j \leq N-1)$ represents the integer $A_{i,j}$ written in the cell in the $(i+1)$-st row from the top and the $(j+1)$-st column from the left of the table A.
  - The argument `B` is a sequence of non-negative integers of length N, where `B[j]` $(0 \leq j \leq N-1)$ represents the integer $B_{r,i,j}$ sent from contestant $(r-1, j)$ to contestant (r, i). If $r = 0$, then `B[j]` $= 0$.
  - This function must return a sequence of non-negative integers of length 1 or N, representing the action taken by contestant (r, i) given the information in the arguments `A`, `B`. If the returned sequence

has length other than 1 or N, the submission is judged as **Wrong Answer [1]**.

- To answer that the value of X is x, this function must return a sequence of length 1, namely (x). If the answered value is incorrect, the submission is judged as **Wrong Answer [2]**.
- If no answer is given, this function must return a sequence B' of length N.
  - For each $j = 0, 1, \ldots, N-1$, $B'[j]$ represents the integer $B_{r+1,j,i}$ sent by contestant (r, i) to contestant $(r+1, j)$. It must be an integer between 0 and $2^{64}-1$, inclusive. Note that even when $r = R-1$ and contestant $(r+1, j)$ does not exist, **the length of B' must still be N**.
- By the end of round $R-1$, at least one contestant must give an answer. If no contestant gives an answer, the submission is judged as **Wrong Answer [3]**.

- **The return value of this function must be determined solely by its arguments.** In particular, note that the return value must not depend on previous calls to `strategy` or on runtime randomness. If this function returns different values for the same arguments, the submission is judged as **Wrong Answer [4]**.
- It is guaranteed that the given arguments can actually occur when playing the game using some table A and the submitted function `strategy`. However, **note that calls are not necessarily made in the order of rounds** $0, 1, \ldots, R-1$.
- **The grader will play one or more games in a single execution.** In one execution, this function is called at most 10 240 times.

Important Notes

- You may freely implement other functions or declare global variables for internal use.
- Your submitted program must not communicate in any way with standard input, standard output, or any other files. However, outputting debugging information and the like to standard error output is allowed.

Constraints

- $2 \leq N \leq 256$.
- $0 \leq A_{i,j} < 2^{48}$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$.
- $A_{i,i} = 0$ $(0 \leq i \leq N-1)$.
- $A_{i,j} = A_{j,i}$ $(0 \leq i < j \leq N-1)$.
- N and $A_{i,j}$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$ are integers.

Subtasks

1. (5 points) $N \leq 64$, $A_{i,j} \leq 1$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$.
2. (10 points) $A_{i,j} \leq 1$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$.
3. (15 points) $N \leq 64$.
4. (40 points) $A_{i,j} < 2^{20}$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$.
5. (15 points) $A_{i,j} < 2^{40}$ $(0 \leq i \leq N-1, 0 \leq j \leq N-1)$.
6. (15 points) There are no additional constraints.

Scoring

If, among the test cases of a subtask, there is even one that is judged as **Wrong Answer [1]–[4]**, Time Limit Exceeded, Memory Limit Exceeded, or Runtime Error, then the score for that subtask is 0. Otherwise, let S be the maximum number of rounds used over all games in that subtask. Then the score for that subtask is determined as follows.

In the case of Subtask 3

- Regardless of the value of S, the score for that subtask is 100% of the points for the subtask. If $S > 6$, the contest site may display "Output is partially correct", but this does not affect the score.

In the case of subtasks other than Subtask 3

- If $S \leq 6$, the score for that subtask is 100% of the points for the subtask.
- If $7 \leq S \leq 9$, the score for that subtask is $(100 - 20 \cdot (S - 6))\%$ of the points for the subtask.
- If $10 \leq S \leq 19$, the score for that subtask is $(40 - S)\%$ of the points for the subtask.
- If $20 \leq S$, the score for that subtask is 20% of the points for the subtask.

Compilation and Test Run

A sample grader for testing your program is included in the archive downloadable from the contest site. This archive also contains a sample file that you must submit.

The sample grader consists of a single file. That file is `grader.cpp`. To test your program, place the files `grader.cpp`, `multi.cpp`, and `multi.h` in the same Presidenty, and execute the following command.

```
g++ -std=gnu++20 -O2 -o grader grader.cpp multi.cpp
```

Alternatively, you may execute the file `compile.sh` included in the archive. In that case, run the following command.

```
./compile.sh
```

If the compilation succeeds, an executable file named `grader` is generated.

Note that the actual grader used for evaluation differs from the sample grader. The sample grader runs as a single process. This program reads input from standard input and writes the results to standard output.

Input for the Sample Grader

The sample grader plays one or more games in a single execution.

First, the sample grader reads the number of games T from standard input. Then, for each of the T games, it reads the information of the table A in the following format.

N
$A_{0,1}\ A_{0,2}\ A_{0,3}\ \cdots\ A_{0,N-1}$
$A_{1,2}\ A_{1,3}\ \cdots\ A_{1,N-1}$
$\vdots$
$A_{N-2,N-1}$

Output for the Sample Grader

The sample grader outputs T lines to standard output. On the t-th line ($1 \leq t \leq T$), it outputs the result of the t-th game in the following format (the quotation marks are not actually printed).

- If the game succeeds, the number of rounds used in that game is output as in "`Accepted: 22`".
- If any of the Wrong Answer conditions applies, the type of Wrong Answer is output as in "`Wrong Answer [4]`".

If the program being executed satisfies more than one Wrong Answer condition, only one of them will be displayed.

Sample Communication

Below is an example of input read by the sample grader and the corresponding sequence of function calls.

Sample Input 1
1
3
1 2
3

Calls to `strategy`	Return Values
`strategy(3, 0, 0, [0, 1, 2], [0, 0, 0])`	`[0, 1, 2]`
`strategy(3, 0, 1, [1, 0, 3], [0, 0, 0])`	`[3, 4, 5]`
`strategy(3, 0, 2, [2, 3, 0], [0, 0, 0])`	`[6, 7, 8]`
`strategy(3, 1, 0, [0, 1, 2], [0, 3, 6])`	`[3]`

In round 0, the following interactions take place.

- Contestant $(0, 0)$ does not answer the value of X, and sends $B_{1,0,0} = 0$ to contestant $(1, 0)$, $B_{1,1,0} = 1$ to contestant $(1, 1)$, and $B_{1,2,0} = 2$ to contestant $(1, 2)$.
- Contestant $(0, 1)$ does not answer the value of X, and sends $B_{1,0,1} = 3$ to contestant $(1, 0)$, $B_{1,1,1} = 4$ to contestant $(1, 1)$, and $B_{1,2,1} = 5$ to contestant $(1, 2)$.
- Contestant $(0, 2)$ does not answer the value of X, and sends $B_{1,0,2} = 6$ to contestant $(1, 0)$, $B_{1,1,2} = 7$ to contestant $(1, 1)$, and $B_{1,2,2} = 8$ to contestant $(1, 2)$.

Since no contestant answered the value of X, the game proceeds to the next round.
In round 1, the following interaction takes place.

- Contestant $(1, 0)$ receives $B_{1,0,0} = 0$ from contestant $(0, 0)$, $B_{1,0,1} = 3$ from contestant $(0, 1)$, and $B_{1,0,2} = 6$ from contestant $(0, 2)$, and answers that $X = 3$. Since the value of X has been answered, the game ends at this point.

Because the answered value of X is correct, the game succeeds. The number of rounds used in this game is 2. This sample input satisfies the constraints of Subtasks $3, 4, 5$, and 6.

Among the files downloadable from the contest site, `sample-01-in.txt` corresponds to Sample Input 1. The archive downloadable from the contest site also contains further sample inputs: `sample-02-in.txt`, `sample-03-in.txt`, and `sample-04-in.txt`. `sample-02-in.txt` satisfies the constraints of all subtasks, `sample-03-in.txt` satisfies the constraints of Subtasks $3, 4, 5$, and 6, and `sample-04-in.txt` satisfies the constraints of Subtasks $4, 5$, and 6.