## Task Prison

⏳2 sec. 💾 1024 MB

Alice and Bob have been unjustly sentenced to a maximum-security prison. Now they must plan their escape. To do this, they need to be able to communicate as efficiently as possible (in particular, Alice needs to send daily information to Bob). However, they cannot meet up and can only exchange information through notes written on napkins. Each day Alice wants to send a new piece of information to Bob – a number between $0$ and $N-1$. At every lunch, Alice gets three napkins and writes a number between $0$ and $M-1$ on each napkin (there may be repetitions) and leaves them on her seat. Then, their enemy, Charly, destroys one of the napkins and mixes the other two up. Finally, Bob finds the two remaining napkins and reads the numbers on them. He must accurately decode the original number that Alice wanted to send him. There is limited space on the napkins, so $M$ is fixed. However, Alice and Bob's goal is to maximize the information throughput, so they are free to choose $N$ as large as they can. Help Alice and Bob by implementing a strategy for each of them, trying to maximize the value of $N$.

## Implementation details

Since this is a communication problem, your program will be run in two separate executions (one for Alice and one for Bob) that cannot share data or communicate in any way other than the one described here. You need to implement three functions:

```
int setup(int M);
```

This will be called once at the start of Alice's execution of your program and once at the start of Bob's execution. It is given $M$ and must return the desired $N$. Both calls to `setup` must return the same $N$.

```
std::vector<int> encode(int A);
```

This implements Alice's strategy. It will be called with the number to encode $A$ ($0 \le A < N$) and must return three numbers $W_1, W_2, W_3$ ($0 \le W_i < M$) that encode $A$. This function will be called a total of $T$ times – once per day (values of $A$ may repeat between days).

```
int decode(int X, int Y);
```

This implements Bob's strategy. It will be called with two of the three numbers returned by `encode` in some order. It must return the same value $A$ that `encode` received. This function will also be called $T$ times – corresponding to the $T$ calls to `encode`; they will be in the same order. All calls to `encode` will happen before all calls to `decode`.

## Constraints

- $M \leq 4300$
- $T = 5000$

## Scoring

For a particular subtask, the fraction S of the points you get depends on the smallest $N$ returned by `setup` on any test in that subtask. It also depends on $N^*$, which is the target value of $N$ that you need to get the full points for the subtask:

- If your solution fails on any test, then $S = 0$.
- If $N \geq N^*$, then $S = 1.0$.
- If $N < N^*$, then $S = \max\left(0.35 \max\left(\frac{\log(N) - 0.985 \log(M)}{\log(N^*) - 0.985 \log(M)}, 0.0\right)^{0.3} + 0.65 \left(\frac{N}{N^*}\right)^{2.4}, 0.01\right)$.

## Subtasks

| **Subtask** | **Points** | $M$ | $N^*$ |
|---|---|---|---|
| 1 | 10 | 700 | 82017 |
| 2 | 10 | 1100 | 202217 |
| 3 | 10 | 1500 | 375751 |
| 4 | 10 | 1900 | 602617 |
| 5 | 10 | 2300 | 882817 |
| 6 | 10 | 2700 | 1216351 |
| 7 | 10 | 3100 | 1603217 |
| 8 | 10 | 3500 | 2043417 |
| 9 | 10 | 3900 | 2536951 |
| 10 | 10 | 4300 | 3083817 |

## Example

Consider the following example with $T = 5$. Here we have an encoding scheme where Alice sends three equal numbers to encode $0$ or three distinct numbers to encode $1$. Notice that Bob can decode the original number from any two of the three numbers Alice sent.

| Execution | Function call | Return value |
|---|---|---|
| Alice | `setup(10)` | `2` |
| Bob | `setup(10)` | `2` |
| Alice | `encode(0)` | `{5, 5, 5}` |
| Alice | `encode(1)` | `{8, 3, 7}` |
| Alice | `encode(1)` | `{0, 3, 1}` |
| Alice | `encode(0)` | `{7, 7, 7}` |
| Alice | `encode(1)` | `{6, 2, 0}` |
| Bob | `decode(5, 5)` | `0` |
| Bob | `decode(8, 7)` | `1` |
| Bob | `decode(3, 0)` | `1` |
| Bob | `decode(7, 7)` | `0` |
| Bob | `decode(2, 0)` | `1` |

## Sample grader

For the sample grader, all calls to `encode` and `decode` will be in the same execution of your program. Additionally, `setup` will be called only once (as opposed to twice, once per execution, as in the grading system).

The input is just a single integer – $M$. Then it will print out the $N$ your `setup` returned. It will then call functions `encode` and `decode` in this order $T$ times with randomly generated numbers from $0$ to $N - 1$ and randomly generated choices of which two of the three numbers from `encode` to give to `decode` (and in what order). It will print out an error message if your solution failed.