Towers

Alice is playing a mobile game where the goal is to protect the base from zombies by placing towers. The base in this game can be represented as a matrix of size $N \times M$. Alice can place towers in any cells that do not go beyond the boundaries of the matrix. The base is considered protected if there is at least one tower in any $K \times K$ square. Help Alice find a tower placement that will protect her base. Since she has just started playing and does not have much money, from all possible placements, output the one that has the minimum number of towers.

Implementation Notes

You must implement the following function:

```
int32 solve(int32 N, int32 M, int32 K)
```

The inputs N, M, K have the same meaning as above, and the function must return the answer specified above.

Note that: the function may be called multiple times in a single execution of the program.

Constraints

- $1 \leq N, M \leq 50.$
- $1 \leq K \leq \min(N, M).$
- The function is called at most $50\,000$ times in a single execution.

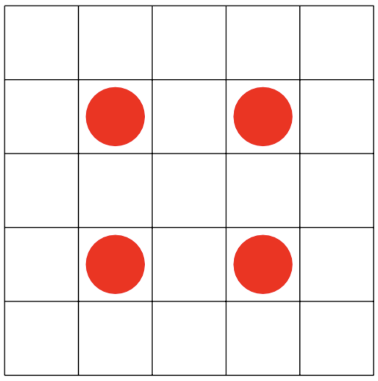
Scoring

1. Subtask 1 (8 points): $K = 1.$
2. Subtask 2 (27 points): $K = 2.$
3. Subtask 3 (31 points): $N = M$ and K divides N.
4. Subtask 4 (34 points): No additional constraints

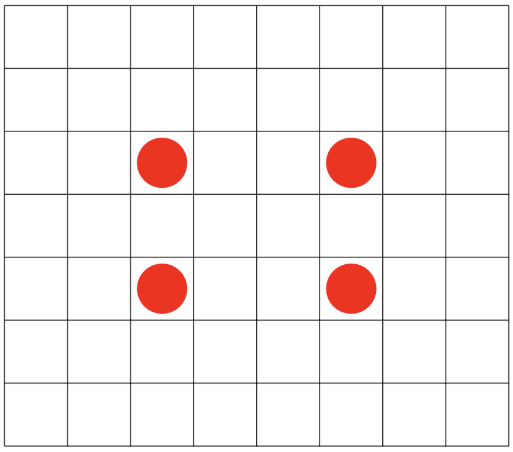
Examples

In the following pictures, towers are red circles.

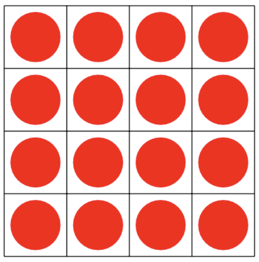
Consider the following call `solve(5, 5, 2)`. In this case ($N = M = 5, K = 2$), and we create the following arrangement:



Consider the following call `solve(7, 8, 3)`. In this case ($N = 7, M = 8, K = 3$), and we create the following arrangement:



Consider the following call `solve(4, 4, 1)`. In this case ($N = M = 4, K = 1$), and we create the following arrangement:



Sample grader

The sample grader reads T, the number of times to call the function `solve`, then T rows, each containing values of N, M, K with which to call `solve`. It then outputs the T output values of `solve` on different lines. The input and output files in the examples use this format.