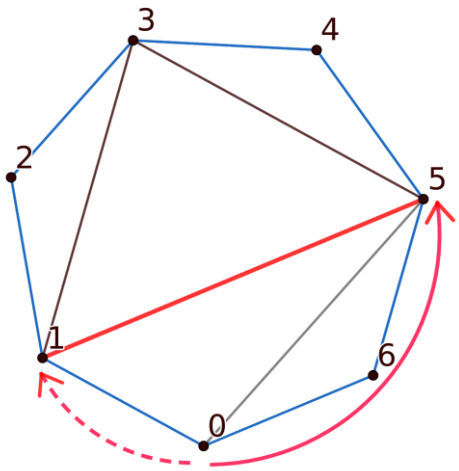
Triangulation

Statement

Anna drew a regular polygon with *n* vertices numbered from 0 to $n-1$ in clockwise order. Later she triangulated it by drawing $n-3$ diagonals that do not intersect each other. Except possibly where they touch at their endpoints. Diagonal is defined as straight lines between two different vertices that aren't sharing a side.

First, let's define the distance from the vertex A to the diagonal D. Suppose we start at vertex A and keep moving to the next vertex in clockwise order until we reach one of the endpoints of D. The number of sides traversed will be called **left_distance**. Similarly, **right_distance** is the number of sides traversed if we start at A and move counter-clockwise until reaching D. The **distance** from A to D is the **maximum** of **left_distance** and **right_distance.**



In the example image the distance from vertex 0 to diagonal (1,5) is 2 with left_distance being 1 and right_distance being 2. As for diagonal (0,5) the distance from vertex 0 is 5, with left_distance=5 and right_distance=2.

Anna wants to make a challenge for Jacob. Jacob does not know which diagonals are drawn at all. He only knows the value of *n*, but can ask Anna multiple times about some pairs of vertices and she will tell him whether a diagonal is present between those vertices. Jacob's goal is to find the closest (with distance defined as above) drawn diagonal from the vertex 0. You are going to help him to achieve his goal by asking Anna a limited amount of questions.

Constraints

- $5 \leq n \leq 100$

Implementation Details

You should implement the following function in your submission:

```
int solve(int n)
```

- This function is called exactly once by the grader
- *n*: number of vertices in the polygon
- This function should return the diagonal between some vertices a and b as an integer with value $a \cdot n + b$
- If there are multiple diagonals with minimal distance you can return any of them

The above function can make calls to the following function:

```
int query(int x, int y)
```

- *x:* the first vertex number
- y: the second vertex number
- $0 \leq x, y \leq n - 1$
- returns 1 if there is a diagonal between x and y and 0 otherwise

Sample interaction

Here is a sample input for grader and corresponding function calls. This input is shown in an image above.
The only line in the input has one integer: *n*
Sample grader will print each query call to the stdout and you have to manually answer it with 1 or 0.

Sample Input to grader	Sample Calls			
	Calls	Returns	Calls	Returns
7	solve(7)			
			query(0, 3)	
				query returns 0
			query(0, 5)	
				query returns 1
			query(1, 5)	
				query returns 1
		solve returns $1 \cdot 7 + 5 = 12$		
		Correct!		

Scoring

Let's denote q as the amount of question you used on a single test. Additionally, $w = \frac{n \cdot (n-3)}{2}$.

- If you ask an invalid question or guess incorrectly you will receive 0% of points for a test
- If $w < q$ you will receive 0% of points for a test
- If $n < q \leq w$ you will receive $10 + 60 \cdot \frac{w-q}{w-n}$% of points for a test
- If $q \leq n$ you will receive 100% of points for a test

Subtasks

There is a single subtask and your score is the sum of individual test scores. But during the contest you will only be able to see scores on half of the tests (worth 50 points). Other half of the scores will be revealed after the contest. Your final score will be **the best total score among all submissions**.