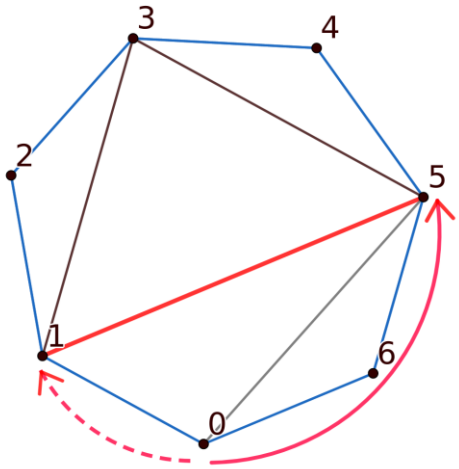
Triangulation

Statement

Anna a desenat un poligon cu *n* vârfuri numerotate de la 0 la *n – 1* în ordinea acelor de ceasornic. Mai târziu a triangulat poligonul desenând *n – 3* diagonale care nu se care nu se intersectează cu excepția posibilă a vârfurilor. O diagonala este o linie dreapta între două vârfuri diferite care nu au o latura între ele.

Să definim mai întâi distanța de la vârful A la diagonala D. Să zicem că începem la vârful A și ne deplasăm la următorul vârf în ordinea acelor de ceasornic pâna ajungem la unul dintre vârfurile lui D. Numărul de muchii traversate il numim **left_distance**. Analog, **right_distance** este numărul de muchii traversate dacă începem de la A și ne deplasăm în direcția opusă acelor ceasului până ajungem la D. Distanța de la A la D este **maximul** dintre **left_distance** și **right_distance.**



În exemplul din imagine distanța de la vârful 0 la diagonala (1, 5) este 2 cu left_distance egal cu 1 și right_distance egal cu 2. Pentru diagonala (0, 5) distanț de la vârful 0 este 5, cu left_distance=5 și right_distance=2.

Anna vrea sa facă din acest lucru o provocare pentru Jacob. Jacob nu știe diagonalele desenate. El știe doar valoarea lui N, dar o poate întreba pe Anna de mai multe ori desepre perechi de vârfuri și ea îi va spune dacă se găsește o muchie între vârfurile acelea. Scopul lui Jacob este să găsească cea mai apropiata (cu noțiunea de distanță de mai sus) diagonala desenata față de vârful 0. Îl veți ajuta să își îndeplinească scopul întrebând-o pe Anna un număr limitat de întrebări.

Constraints

- $5 \leq n \leq 100$

Implementation Details

Trebuie să implementați următoarea funcție în submisia voastră:

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

- Funcția se apelează exact odată de către evaluator.
- n: numărul de vârfuri din poligon.
- Funcția ar trebui să returneze diagonala între niște vârfuri a și b ca un întreg cu valoarea $a \cdot n + b$
- Dacă sunt mai multe diagonale cu aceeași distanță minimă, puteți să returnați oricare.

Funcția de mai sus poate apela următoarea funcție:

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

- x: indicele primului vârf
- y: indicele celui de al doilea vârf
- $0 \leq x, y \leq n$
- returnează 1 dacă există o diagonală între x și y și 0 altfel.

Sample interaction

Urmează un exemplu de input pentru evaluator și apelurile de funcții corespunzătoare. Inputul este cel desenat in imaginea de mai sus.
Singura linie din input corespunde lui n.
Evaluatorul va afișa fiecare apel al lui query la stdout și trebuie să îi răspundeți manual cu 1 sau 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

Fie q numărul de interogări ce le faceți pe un singur test. Totodată fie $w = \frac{n \cdot (n-3)}{2}$.

- Dacă faci o interogare invalidă sau ghiciți răspunsul incorect veți primi 0% din puncte.
- Dacă $w < q$ veți primi 0% din puncte pentru test.
- Dacă $n < q \leq w$ veți primi $10 + 60 \cdot \frac{w-q}{w-n}$% din puncte pentru un test.
- Dacă $q \leq n$ veți primi 100% din puncte pe test.

Subtasks

Va fi un singur subtask iar scorul vostru este suma scorurilor pe testele individuale. Dar, pe parcursul concursului veți putea vedea scorurile doar pe jumătate din teste (ce valorează 50 de puncte). Celelalte scoruri se vor afișa după concurs. Scorul final va fi **scorul maxim dintre toate submisiile.**