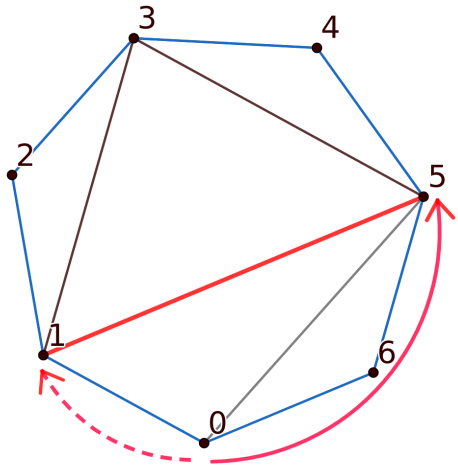
Тріангуляція

Умова

Анна намалювала правильний многокутник з n вершинами, пронумерованими від 0 до $n-1$ за годинниковою стрілкою. Потім вона тріангулювала його, намалювавши $n-3$ діагоналі, які не перетинають одна одну, за винятком, можливо, місця, де вони торкаються своїх кінцевих точок. Діагональ – це пряма лінія між двома різними вершинами, які не мають спільної сторони.



Спочатку визначимо відстань від вершини A до діагоналі D. Припустимо, ми починаємо з вершини A і продовжуємо рухатися до наступної вершини за годинниковою стрілкою, поки не дійдемо до однієї з кінцевих точок D. Кількість пройдених сторін буде називатися **left_distance**. Подібним чином, **right_distance** – це кількість сторін, пройдених, якщо ми починаємо з вершини A і рухаємось проти годинникової стрілки, поки не досягнемо D. **Відстань** від A до D – це **максимальна** відстань з двох відстаней: **left_distance** та **right_distance**.

На рисунку відстань від вершини 0 до діагоналі (1,5) дорівнює 2, при цьому left_distance дорівнює 1, а right_distance дорівнює 2. А до діагоналі (0,5) відстань від вершини 0 дорівнює 5, при цьому left_distance дорівнює 5 і right_distance дорівнює 2

Анна хоче придумати задачку для Якова. Яків не знає, які діагоналі проведені. Він знає лише значення n і може кілька разів запитати Анну про деякі пари вершин, і вона скаже йому, чи існує діагональ між цими вершинами. Яків бажає знайти найближчу (з відстанню, визначеною вище) намальовану діагональ від вершини 0. Ви збираєтеся допомогти йому досягти своєї мети, поставивши Анні обмежену кількість питань.

Обмеження

- $5 \leq n \leq 100$

Деталі реалізації

Ви маєте реалізувати наступну функцію:

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

- Ця функція буде викликана рівно один раз грейдером
- *n*: кількість вершин многокутника
- Ця функція повинна повертати діагональ між деякими вершинами a і b як ціле число зі значенням $a \cdot n + b$.
- Якщо є декілька діагоналей з мінімальною відстанню, ви можете повернути будь-яку з них.

Вищезазначена функція може здійснювати виклики до наступної функції:

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

- *x:* номер першої вершини
- y: номер другої вершини
- $0 \leq x, y \leq n - 1$
- повертає 1, якщо між x та y є діагональ, та повертає 0 в іншому випадку

Приклад взаємодії

Приклад вхідних даних для грейдера та відповідних викликів функцій. Цей приклад також зображений на рисунку вище.
Єдиний рядок вхідних даних містить n.
Грейдер виводить кожен виклик функції query в потік вихідних даних (stdout), і ви повинні відповідати на нього вручну: 1 або 0.

Приклад вхідних даних для грейдера	Приклади викликів функцій			
	Виклик	Результат	Виклик	Результат
7	solve(7)			
			query(0, 3)	
				query повертає 0
			query(0, 5)	
				query повертає 1
			query(1, 5)	
				query повертає 1
		solve повертає $1 \cdot 7 + 5 = 12$		
		Правильна відповідь!		

Оцінювання

Позначимо q як кількість запитань, яку ви використали під час одного тесту. Крім того, $w = \frac{n \cdot (n-3)}{2}$.

- Якщо ви поставите неправильне запитання або відповісте неправильно, ви отримаєте 0% балів за тест
- Якщо $w < q$ ви отримаєте 0% балів за тест
- Якщо $n < q \leq w$ ви отримаєте $10 + 60 \cdot \frac{w-q}{w-n}$% балів за тест
- Якщо $q \leq n$ ви отримаєте 100% балів за тест

Блоки

Існує один блок, і ваш бал – це сума результатів кожного тесту окремо. Але під час змагання ви зможете бачити бали лише за першу половину тестів (вартістю 50 балів). Результати роботи програми на іншій половині тестів будуть розкриті тільки після змагання. Остаточний результат – це **найкращий загальний бал серед усіх зроблених спроб**.