Chika Wants to Cheat

Задача	Cheat
Вхідні дані	Інтерактивна задача
Вихідні дані	Інтерактивна задача
Ліміт часу	2 секунди
Ліміт пам'яті	512 МБ

Чіка має колоду з q гральних карт, пронумерованих різними натуральними числами. Вона хоче пограти в ігри зі своїми друзями зі студентської ради академії Shuchi'in, використовуючи ці карти. Але вона також хоче виграти, тому вона вирішує таємно позначити зворотний бік карт у своїй колоді.

Усі карти є квадратної форми та розміром 2×2, де нижній лівий кут має координати $(0, 0)$, а верхній правий кут має координати $(2, 2)$. Чіка малює на зворотному боці кожної карти певний візерунок, щоб потім, дивлячись на нього, знати, яке число написано на лицьовій стороні картки. Вона малює такий візерунок за такою процедурою: будь-яку кількість разів (можливо, 0), вона вибирає дві різні точки A і B, які мають цілі координати відносно нижнього лівого кута карти, і малює між ними **прямий відрізок**.

Чіка буде малювати лише **правильні** відрізки, тобто відрізки між двома точками A і B, для яких немає іншої точки C (відмінної від A і B) з цілими координатами, яка також лежить на відрізку. Наприклад, відрізок між $(0, 0)$ і $(2, 2)$ є **неправильним**, оскільки містить точку $(1, 1)$, а відрізки між $(0, 0)$ і $(1, 1)$ та між $(1, 1)$ і $(2, 2)$ обидва є **правильними**, і Чіка навіть може намалювати їх обидвох на одному візерунку. Також зауважте, що відрізки не мають напрямку: відрізок, проведений від A до B, **ідентичний** як собі, так і відрізку, проведеному у зворотному напрямку, від B до A.

Важливо те, що Чіка хоче бути впевнена, що вона розпізнає свої карти незалежно від того, як вони будуть повернуті. Карту можна повернути на 0, 90, 180 або 270 градусів проти годинникової стрілки відносно початкової орієнтації.

Ваше завдання — допомогти Чіці розробити візерунки для q карт у її колоді, а потім розпізнати ці карти пізніше.

Реалізація

*Це інтерактивне завдання з двох етапів, **<u>кожний етап передбачає окремий запуск вашої програми</u>**.* Вам потрібно реалізувати дві функції:

- Функція `BuildPattern`, яка повертає візерунок, який слід намалювати ззаді певної карти. Ця функція буде викликана q раз під час першого запуску.
- Функція `GetCardNumber`, яка повертає номер (ймовірно повернутої) карти, яка має намальований ззаді певний візерунок. Ця функція буде викликана q разів під час другого запуску.

Перша функція

```
std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> BuildPattern(int n);
```

приймає єдиний параметр n, число, написане на лицьовій стороні карти. Вам потрібно повернути `std::vector`, що містить відрізки, які Чіка малює як візерунок на звороті карти, щоб розпізнати його пізніше. Відрізок представлено як `std::pair` точок, а точка представлена як `std::pair` (x, y) цілих координат відносно лівого нижнього кута карти (який має координати $(0, 0)$), де $0 \leq x, y \leq 2$. Усі відрізки, які малює Чіка, мають бути правильними та попарно різними. Гарантується, що всі q викликів до `BuildPattern` отримують різні параметри n.

Після отримання всіх візерунків для q карт, грейдер може виконати будь-яку з наступних операцій будь-яку кількість разів для кожного візерунка:

- Повернути весь візерунок на 0, 90, 180 або 270 градусів проти годинникової стрілки.
- Змінити порядок відрізків у представленні візерунка через `std::vector`.
- Змінити місцями кінцеві точки відрізка у візерунці. (Відрізок, проведений від A до B, може стати таким самим відрізком від B до A.)

Друга фукція

```
int GetCardNumber(std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> p);
```

приймає один параметр p, `std::vector` відрізків, що описують візерунок, намальований Чікою на звороті карти, на основі значення, поверненого викликом вашої функції `BuildPattern`. Функція має повернути число n, написане на лицьовій стороні карти. Пам'ятайте, що візерунок p не обов'язково має ту форму, яку повертає `BuildPattern`; він міг бути змінений трьома вище згаданими операціями. Також можливо, що порядок карт відрізняється від порядку, в якому вони були подані під час першого етапу, але гарантовано, що кожна карта буде використана рівно один раз.

Обмеження

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ для всіх викликів функції `BuildPattern`.
- Зауважте, що існує алгоритм для побудови візерунків, який дозволяє розпізнавати до $67\,000\,000$ різних карт.

Оцінювання

- Підзадача 1 (2 бали): $n \leq 2$.
- Підзадача 2 (9 балів): $n \leq 25$.
- Підзадача 3 (15 балів): $n \leq 1\,000$ і грейдер **не буде повертати** картки між першим та другим етапами. (Грейдер **може** виконувати дві інші операції.)
- Підзадача 4 (3 бали): $n \leq 16\,000\,000$ і грейдер **не буде повертати** картки між першим та другим етапами. (Грейдер **може** виконувати дві інші операції.)
- Підзадача 5 (24 бали): $n \leq 16\,000\,000$.
- Підзадача 6 (18 балів): $n \leq 40\,000\,000$.
- Підзадача 7 (29 балів): Без додаткових обмежень.

Приклад взаємодії

Викликана функція	Повернене значення	Пояснення
Початок першого етапу.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Нам треба створити візерунок для числа 3 на карті розміру 2×2. Ми вирішили намалювати 2 відрізки: - між $(0, 0)$ та $(2, 1)$, - між $(1, 1)$ та $(2, 0)$.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Нам треба створити візерунок для числа 1 на карті розміру 2×2. Ми вирішили намалювати 1 відрізок: - між $(0, 1)$ та $(0, 0)$.
Завершення першого етапу.	-	-
Початок другого етапу.	-	-
`GetCardNumber( {{{0, 0}, {0, 1}}})`	`1`	Ми отримали візерунок, що складається з 1 відрізка. - між $(0, 0)$ та $(0, 1)$. Це є той самий візерунок, який ми б отримали, якщо б намалювали відрізок: - між $(0, 1)$ та $(0, 0)$,

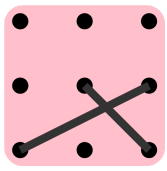
		що є тим самим візерунком, з такою ж самою орієнтацією (візерунок повернений на 0 градусів), який ми повернули під час другого виклику функції `BuildPattern`. Отже, повертаємо 1.

`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Ми отримали візерунок, що складається з 2-х відрізків:
		- між $(1, 1)$ та $(2, 2)$,
		- між $(1, 2)$ та $(2, 0)$.
		Це є візерунок, який ми повернули під час першого виклику функції `BuildPattern`, повернений на 90 градусів проти годинникової стрілки. Отже, повертаємо 3.

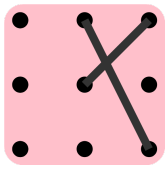
Завершення другого етапу.	-	-

Наступні три зображення представляють по порядку:

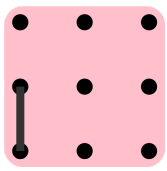
- Візерунок, повернений під час першого виклику `BuildPattern`:



- Візерунок, переданий як параметр під час другого виклику `GetCardNumber`, який є першим шаблоном поверненим на 90 градусів проти годинникової стрілки.



- Візерунок, отриманий під час другого виклику `BuildPattern`, який також є параметром під час першого виклику `GetCardNumber`.



Зразок грейдера

Наданий зразок грейдера, `grader.cpp`, у вкладенні задачі `Cheat.zip`. Грейдер зчитує ціле число q зі стандартного введення, а потім виконує наступні кроки q разів:

- Ввід цілого числа n зі стандартного введення.
- Виклик `BuildPattern(n)` і збереження повернутого значення в змінній p.
- Виклик `GetCardNumber(p)` і виведення поверненого значення у стандартний вивід.

Ви можете змінити свій грейдер локально, якщо хочете.

Щоб скомпілювати зразок грейдера з вашим рішенням, ви можете використати наступну команду в терміналі:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp
```

де `solution.cpp` це ваше рішення, яке ви відправляєте у CMS. Щоб запустити програму з прикладом вхідних даних, наданим у вкладенні, введіть таку команду в термінал:

```
./solution < input.txt
```

Зверніть увагу, що, на відміну від зразка грейдера, справжній грейдер на CMS виконуватиме перший і другий етапи в окремих запусках вашої програми.