Chika Wants to Cheat

Nombre del problema	Cheat
Fichero de entrada	Problema interactivo
Fichero de salida	Problema interactivo
Límite de tiempo	2 segundos
Límite de memoria	512 megabytes

Chika tiene una baraja de q cartas, numeradas con varios enteros positivos. Quiere jugar algunas partidas con sus amigos de la asociación estudiantil de la Academia Shuchi'in usando estas cartas, pero también quiere ganar, así que decide marcar secretamente las partes de atrás de las cartas de su baraja.

Las cartas todas tienen forma de cuadrado y tamaño 2×2, donde la casilla de abajo a la izquierda tiene coordenadas $(0, 0)$ y la casilla de arriba a la derecha tiene coordenadas $(2, 2)$. Chika dibuja un patrón concreto en la parte de atrás de cada carta, de forma que más tarde sabrá, mirando al patrón, qué número hay en la cara delantera de la carta. Dibuja tal patrón usando el siguiente método: tantas veces como quiera (posiblemente 0 veces), escogerá dos puntos distintos A y B (cuyas coordenadas siempre serán números enteros en relación con la coordenada inferior izquierda de la carta) y dibuja una **segmento de línea recta** entre ellos.

Sin embargo, Chika dibujará únicamente segmentos **válidos**, es decir, entre dos puntos A y B para los que no hay otro punto C (diferente de A y B) con coordenadas que sean números enteros y que también esté en el segmento. Por ejemplo, el segmento entre $(0, 0)$ y $(2, 2)$ es **inválido**, al contener el punto $(1, 1)$, pero los segmentos entre $(0, 0)$ y $(1, 1)$ y entre $(1, 1)$ y $(2, 2)$ son ambos **válidos**, y Chika puede incluso dibujar ambos en el mismo patrón. Además, ten en cuenta que los segmentos no tienen dirección: un segmento dibujado de A a B es **idéntico** tanto a sí mismo como a uno dibujado en la dirección opuesta, de B a A.

Sobre todo, Chika quiere asegurarse de que va a reconocer sus cartas independientemente de cómo estén rotadas. Una carta puede estar rotada 0, 90, 180 o 270 grados en sentido contrario a las agujas del reloj con respecto a la orientación original.

Tu tarea será ayudar a Chika a diseñar los patrones para las q cartas de su baraja y ayudarla a reconocerlas más adelante.

Implementación

Este es un problema interactivo con dos fases, ***cada fase ejecutándose en una ejecución distinta de tu programa***. Has de implementar dos funciones:

- Una función `BuildPattern` que devuelve el patrón a dibujar en la parte de atrás de una carta dada. Esta función será llamada q veces en la primera fase.
- Una función `GetCardNumber` que devuelve el número de una (posiblemente rotada) carta que tiene un patrón dibujado en la primera fase. Esta función será llamada q veces en la segunda fase.

La primera función

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

recibe un único parámetro n, el número escrito en la cara delantera de la carta. Has de devolver un `std::vector` que contenga los segmentos que Chika dibuja como un patrón en la parte trasera de la carta para reconocerlo más adelante. Un segmento estará representado como una `std::pair` de puntos, y un punto estará representado como una `std::pair` (x, y) de coordenadas que sean números enteros en relación con la esquina inferior izquierda de la carta, donde $0 \leq x, y \leq 2$. Todos los segmentos que Chika dibuje han de ser válidos y no podrá haber ninguna pareja de segmentos idénticos. Se garantiza que las q llamadas a BuildPattern recibirán diferentes valores para el parámetro n.

Tras recibir todos los patrones para las q cartas, el corrector puede realizar cualquiera de las siguientes operaciones, cualquier número de veces, en cada uno de los patrones:

- Rotar el patrón completo 0, 90, 180 o 270 grados en sentido contrario a las agujas del reloj.
- Modificar el orden de los segmentos en la representación del patrón en forma de `std::vector`.
- Cambiar el orden de los puntos de un segmento en el patrón. (Un segmento dibujado de A a B puede volverse el segmento idéntico de B a A.)

La segunda función,

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

recibe un único parámetro p, un `std::vector` de segmentos describiendo el patrón dibujado por Chika en la parte de atrás de su carta, basado en el valor devuelto de una llamada anterior a tu función `BuildPattern`. La función ha de devolver el número n dibujado en la parte delantera de la carta. Recuerda que el patrón p no está necesariamente en el formato original devuelto por `BuildPattern`; puede haber sido sometido a las tres operaciones mencionadas anteriormente.

También es posible que el orden de las cartas sea diferente del orden en el que fueron recibidas en la primera fase, pero se garantiza que cada carta se usará como mucho una vez.

Restricciones

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ para todas las llamadas a la función `BuildPattern`.
- Ten en cuenta que existen algoritmos para construir patrones de forma que uno pueda reconocer $67\,000\,000$ cartas diferentes.

Puntuaciones

- Subtarea 1 (2 puntos): $n \leq 2$.
- Subtarea 2 (9 puntos): $n \leq 25$.
- Subtarea 3 (15 puntos): $n \leq 1\,000$ y el corrector **no rotará** los patrones entre las fases 1 y 2. (El corrector **podrá** realizar las otras dos operaciones.)
- Subtarea 4 (3 puntos): $n \leq 16\,000\,000$ y el corrector **no rotará** los patrones entre las fases 1 y 2. (El corrector **podrá** realizar las otras dos operaciones.)
- Subtarea 5 (24 puntos): $n \leq 16\,000\,000$.
- Subtarea 6 (18 puntos): $n \leq 40\,000\,000$.
- Subtarea 7 (29 puntos): Sin restricciones adicionales.

Ejemplo de interacción

Llamada a la función	Valor devuelto	Explicación
La primera ejecución comienza.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Tenemos que crear un patrón para el número 3 en la carta de tamaño 2×2. Decidimos dibujar 2 segmentos: - entre $(0, 0)$ y $(2, 1)$, - entre $(1, 1)$ y $(2, 0)$.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Tenemos que crear un patrón para el número 1 en la carta de tamaño 2×2. Decidimos dibujar 1 segmento: - entre $(0, 1)$ y $(0, 0)$.
La primera ejecución termina.	-	-
La segunda ejecución comienza.	-	-
`GetCardNumber(`	`1`	Recibimos un patrón compuesto por 1

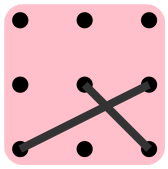
`{{{0, 0}, {0, 1}}})`		segmento: - entre $(0,0)$ y $(0,1)$. Este es exactamente el mismo patrón que obtendríamos al dibujar el segmento: - entre $(0,1)$ y $(0,0)$ que es exactamente el mismo patrón con la misma orientación (rotado 0 grados) que devolvimos en la segunda llamada a la función `BuildPattern`. Por lo tanto, retornamos 1.

`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Recibimos un patrón compuesto por 2 segmentos:
		- entre $(1,1)$ y $(2,2)$,
		- entre $(1,2)$ y $(2,0)$.
		Este es el patrón que devolvimos en la primera llamada a la función `BuildPattern`, rotado 90 grados en sentido contrario a las agujas del reloj. Por lo tanto, devolvemos 3.

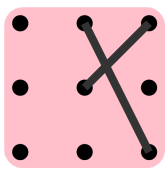
La segunda ejecución termina.	-	-

Los siguientes tres diagramas representan, en orden:

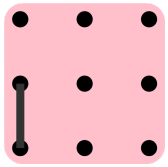
- El patrón devuelto como salida por la primera llamada a `BuildPattern`:



- El patrón recibido como parámetro por la segunda llamada a `GetCardNumber`, que es el primer patrón tras rotarlo 90 grados en sentido contrario a las agujas del reloj.



- El patrón devuelto como salida por la segunda llamada a `BuildPattern`, que es a su vez el mismo patrón recibido como argumento en la primera llamada a `GetCardNumber`.



Corrector de ejemplo

El corrector de ejemplo proporcionado,`grader.cpp`, en el adjunto `Cheat.zip`, lee un entero q de la entrada estándar y luego realiza los siguientes pasos q veces:

- Leer un entero n de la entrada estándar.
- Llamar a `BuildPattern(n)` y guardar el valor devuelto en una variable p.
- Llamar a `GetCardNumber(p)` e imprimir el valor devuelto a la entrada estándar.

Puedes modificar tu corrector localmente si así lo deseas.

Para compilar el corrector de ejemplo con tu solución, puedes usar el siguiente comando en la terminal:

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

donde `solution.cpp` es tu archivo con la solución a enviar a CMS. Para ejecutar tu programa con la entrada de ejemplo proporcionada en el adjunto, teclea el siguiente comando en la terminal:

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

Por favor ten en cuenta que, al contrario que el corrector de ejemplo, el corrector de verdad en CMS realizará la primera fase y la segunda fase en ejecuciones distintas de tu programa.