Chika Wants to Cheat

Problem Name	Cheat
Input File	Interactive task
Output File	Interactive task
Time limit	2 seconds
Memory limit	512 megabytes

Chika are un pachet de q cărți de joc numerotate cu diferite numere întregi pozitive. Ea vrea să joace niște jocuri cu prietenii ei din consiliul studenților de la Shuchi'in Academy utilizând aceste cărți, dar ea vrea să și câștige, deci ea decide să marcheze în secret spatele cărților din pachetul său.

Toate cărțile au formă pătrată de 2×2, unde colțul stânga-jos are coordonatele $(0, 0)$ și colțul dreapta-sus are coordonatele $(2, 2)$. Chika desenează un anumit model pe spatele fiecărei cărți, astfel încât ea să știe mai târziu, uitându-se la model, ce număr este scris pe fața cărții de joc. Ea desenează un astfel de model utilizând următoarea procedură: Ori de câte ori dorește (posibil de 0 ori), ea alege două puncte distincte A și B care au coordonate întregi relativ la colțul de stânga-jos al cărții și desenează un **segment de linie dreaptă** între ele.

Chika va desena numai segmente **valide**, adică, segmente între două puncte A și B pentru care nu există un alt punct C (diferit de A și B) de coordonate întregi care aparține de asemenea segmentului. De exemplu, segmentul dintre $(0, 0)$ și $(2, 2)$ **nu este valid** deoarece conține punctul $(1, 1)$, dar segmentele dintre $(0, 0)$ și $(1, 1)$ și dintre $(1, 1)$ și $(2, 2)$ sunt ambele **valide** și Chika poate chiar să le deseneze pe ambele în același model. De asemenea, rețineți că segmentele nu au o direcție: un segment desenat de la A la B este **identic** cu el însuși și cu segmentul desenat în direcție inversă, de la B la A.

Important, Chika vrea să fie sigură că ea își va recunoaște cărțile indiferent de cum sunt rotite. O carte poate fi rotită cu 0, 90, 180 sau 270 grade în sens invers acelor de ceasornic referitor la orientarea lor originală.

Sarcina ta este să o ajuți pe Chika să proiecteze modelele pentru cele q cărți din pachetul ei și să recunoască ulterior acele cărți.

Implementare

*Aceasta este o problemă interactivă cu două etape, **fiecare etapă implicând o execuție separată a programului tău***. Tu trebuie să implementezi două funcții:

- O funcție `BuildPattern` care returnează modelul care să fie desenat pe o carte dată. Această funcție va fi apelată de q ori în prima etapă.
- O funcție `GetCardNumber` care returnează numărul unei cărți (posibil rotită) care are un anumit model desenat în prima etapă. Această funcție va fi apelată de q ori în a doua etapă.

Prima funcție

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

primește un singur parametru n, numărul care este scris pe fața cărții de joc. Tu trebuie să returnezi un `std::vector` conținând segmentele pe care Chika le desenează ca model pe spatele cărții pentru a o recunoaște ulterior. Un segment este reprezentat ca un `std::pair` de puncte și un punct este reprezentat ca un `std::pair` (x, y) de coordonate întregi relative la colțul stânga-jos al cărții de joc, unde $0 \leq x, y \leq 2$. Toate segmentele pe care Chika le desenează trebuie să fie valide și neidentice două câte două. Se garantează că toate cele q apeluri la BuildPattern primesc valori diferite pentru parametrul n.

După primirea modelelor pentru toate cele q cărți, evaluatorul poate face oricare dintre următoarele operații, de ori câte ori, pentru fiecare dintre modele:

- Rotește întregul model cu 0, 90, 180 sau 270 grade în sens invers acelor de ceasornic.
- Modifică ordinea segmentelor în reprezentarea modelului din `std::vector`.
- Schimbă ordinea extremităților unui segment din model (un segment desenat de la A la B poate deveni identic cu segmentul de la B la A.)

A doua funcție,

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

primește un singur parametru p, un `std::vector` de segmente descriind modelul care este desenat de Chika pe spatele cărții, bazat pe valoarea returnată de un apel precedent al funcției tale `BuildPattern`. Funcția trebuie să returneze numărul n scris pe fața cărții de joc. Amintește-ți că modelul p nu este în mod necesar în forma originală care a fost returnată de `BuildPattern`; el este posibil să fi fost supus celor trei operații menționate mai sus. Este de asemenea posibil ca ordinea cărților să fie diferită de ordinea în care au fost date în prima etapă, dar se garantează ca fiecare carte va fi utilizată exact o dată.

Restricții

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ pentru toate apelurile funcției `BuildPattern`.
- Rețineți că există algoritmi care să construiască modele astfel încât să poată fi recunoscute $67\,000\,000$ de cărți diferite.

Punctaj

- Subtask 1 (2 puncte): $n \leq 2$.
- Subtask 2 (9 puncte): $n \leq 25$.
- Subtask 3 (15 puncte): $n \leq 1\,000$ și evaluatorul **nu va roti** modelele între etapele 1 și 2. (Evaluatorul **poate** efectua celelalte două operații.)
- Subtask 4 (3 puncte): $n \leq 16\,000\,000$ și evaluatorul **nu va roti** modelele între etapele 1 și 2. (Evaluatorul **poate** efectua celelalte două operații.)
- Subtask 5 (24 puncte): $n \leq 16\,000\,000$.
- Subtask 6 (18 puncte): $n \leq 40\,000\,000$.
- Subtask 7 (29 puncte): Nu există restricții suplimentare.

Exemplu de interacțiune

Apel de funcție	Valoare returnată	Explicație
Începe prima etapă.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Trebuie să creăm un model pentru numărul 3 pe o carte de dimensiuni 2×2. Decidem să desenăm 2 segmente: - între $(0, 0)$ și $(2, 1)$, - între $(1, 1)$ și $(2, 0)$.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Trebuie să creăm un model pentru numărul 1 pe o carte de dimensiuni 2×2. Decidem să desenăm 1 segment: - între $(0, 1)$ și $(0, 0)$.
Prima etapă s-a terminat.	-	-
Începe a doua etapă.	-	-
`GetCardNumber({{{0, 0}, {0, 1}}})`	`1`	Obținem un model făcut din 1 segment: - între $(0, 0)$ și $(0, 1)$. Acesta este același model pe care l-am obține prin desenarea segmentului: - între $(0, 1)$ și $(0, 0)$

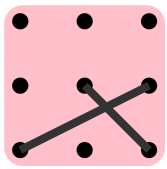
		care este exact același model cu aceeași orientare (rotit cu 0 grade) pe care l-am returnat la al doilea apel al funcției `BuildPattern`. Prin urmare, returnăm 1.

`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Obținem un model format din 2 segmente:
		- între $(1, 1)$ și $(2, 2)$,
		- între $(1, 2)$ și $(2, 0)$.
		Acest model l-am returnat la primul apel al funcției `BuildPattern`, rotit cu 90 de grade în sensul invers al acelor de ceasornic. Prin urmare, returnăm 3.

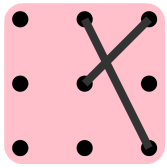
A doua etapă se termină.	-	-

Următoarele trei imagini reprezintă, în ordine:

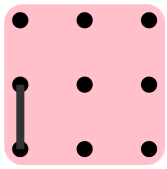
- Modelul returnat ca ieșire a primului apel al `BuildPattern`:



- Modelul primit ca parametru de al doilea apel al `GetCardNumber`, care este primul model rotit cu 90 de grade în sensul invers acelor de ceasornic.



- Modelul returnat ca ieșire de al doilea apel al `BuildPattern`, care este de asemenea modelul primit ca parametru de primul apel al `GetCardNumber`.



Grader exemplu

Grader-ul exemplu furnizat, `grader.cpp`, în atașamentul problemei `Cheat.zip`, citește un întreg q de la intrarea standard și apoi va face următorii pași de q ori:

- Citește un întreg n de la intrarea standard.
- Apelează `BuildPattern(n)` și memorează valoarea returnată într-o variabilă p.
- Apelează `GetCardNumber(p)` și afișează valoarea returnată la ieșirea standard.

Puteți modifica grader-ul local dacă doriți acest lucru.

Pentru a compila grader-ul exemplu cu soluția ta, poți utiliza următoarea comandă la promptul terminalului:

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

unde `solution.cpp` este fișierul cu soluția ta care va fi trimis pe CMS. Pentru a executa programul cu intrarea exemplu furnizată în atașament, tastați următoarea comandă la promptul terminalului:

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

Te rog să observi că, spre deosebire de grader-ul exemplu, grader-ul real de pe CMS va executa prima etapă și a doua etapă în execuții separate ale programului tău.