Chika Wants to Cheat

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

Chika má q hracích kariet. Zozadu sú čisté, spredu má každá na sebe iné rozumne malé prirodzené číslo.

Chika si chce zahrať karty s účastníčkami EGOI. Keďže nerada prehráva, rozhodla sa, že bude podvádzať: na každú kartu si zozadu namaľuje iný vzor, a to tak, aby pre každú kartu vedela zo zadnej strany určiť, aké číslo je na prednej strane.

Všetky karty majú tvar štvorca rozmerov 2×2. Ak si na karte zavedieme súradnicový systém začínajúci v ľavom dolnom rohu, bude na nej práve deväť bodov, ktoré majú obe súradnice celočíselné: od $(0, 0)$ vľavo dole po $(2, 2)$ vpravo hore. Tieto budeme volať *významné body*.

Platná úsečka je úsečka, ktorá spája dva navzájom rôzne významné body a neobsahuje na sebe žiadny iný významný bod. Napr. úsečka spájajúca dva protiľahlé rohy karty nie je platná, lebo v jej strede leží významný bod $(1, 1)$. Ale napríklad už úsečka medzi $(0, 0)$ a $(1, 1)$ a aj úsečka medzi $(1, 1)$ a $(2, 2)$ platné sú.

Na každej karte Chika nakreslí nejaký vzor tak, že postupne koľkokrát chce (aj nulakrát) nakreslí nejakú novú platnú úsečku. Keby napríklad nakreslila obe platné úsečky spomenuté v predchádzajúcom odseku, dostala by kartu, na ktorej je nakreslená celá uhlopriečka z rohu $(0, 0)$ do rohu $(2, 2)$.

Dve úsečky voláme *identické* ak spájajú tú istú neusporiadanú dvojicu bodov.

Teraz prichádza hlavná pointa celej úlohy: karty sú štvorcové a počas hry sa ľahko môže stať, že budú otočené o nejaký násobok 90 stupňov. Chika potrebuje vzory kresliť tak, aby vedela každú kartu spoznať bez ohľadu na to, ako je otočená.

Tvojou úlohou je povedať jej, ako kresliť. Tvoj program musí najskôr navrhnúť vzory pre všetkých q kariet, ktoré Chika má, a potom prejsť testom, ktorý overí, že vieš z tvojich vzorov správne zistiť

čísla na jednotlivých kartách.

Implementation

Toto je interaktívna úloha s dvoma postupnými fázami. ***Každej fáze bude zodpovedať samostatné spustenie tvojho programu.***.

Potrebuješ implementovať dve funkcie:

- Funkciu `BuildPattern`, ktorá pokreslí jednu danú kartu. Túto funkciu v prvej fáze grader q-krát zavolá a tým postupne získa balíček q pokreslených kariet.
- Funkciu `GetCardNumber`, ktorá dostane popis vzoru na jednej z tvojich kariet (možno otočenej) a má povedať, aké číslo je na nej. Túto funkciu grader q-krát zavolá v druhej fáze a tým skontroluje, či všetky karty rozoznáš správne.

Detaily o prvej funkcii:

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

Táto funkcia má jediný parameter: kladné celé číslo n, napísané na karte, ktorú máš pokresliť.

Je zaručené, že všetky hodnoty n použité v postupných volaniach tejto funkcie budú navzájom rôzne. V každej podúlohe je daný nejaký rozsah, v ktorom všetky použité n ležia.

Na výstupe má táto funkcia vrátiť vektor, ktorého prvky sú popisy jednotlivých úsečiek, ktoré chceš nakresliť. Každý z týchto popisov je `std::pair`, ktorého prvky sú dva body: koncové body úsečky. Každý bod je `std::pair` súradníc.

Všetky úsečky musia byť platné a žiadne dve nesmú byť identické.

Po skončení volaní prvej funkcie:

Keď grader postupne získa vzory na všetkých q kartách, môže si ich upraviť, a to tak, že ľubovoľne veľakrát spraví s každým vzorom ľubovoľnú z nasledujúcich operácií:

- Otočí ho celý o 0, 90, 180 alebo 270 stupňov.
- Zmení poradie úsečiek vo vektore, ktorý ho popisuje.
- Nahradí niektorú úsečku v tom vektore ekvivalentnou, ktorá má vymenené poradie koncových bodov. (Teda namiesto úsečky AB tam dá úsečku BA.)

Detaily o druhej funkcii:

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

Jediným parametrom tejto funkcie je popis vzoru presne v takom formáte, v akom ho vracia funkcia `BuildPattern`. Táto funkcia z neho musí vypočítať a následne vrátiť číslo n, ktoré je na karte s týmto vzorom.

Pripomíname, že vzor, ktorý táto funkcia dostane na vstupe, vznikol z jedného z tvojich q vzorov postupnosťou vyššie uvedených operácií.

Pre každú z tvojich q kariet spraví grader jedno volanie tejto funkcie. Nemusí to však robiť v tom istom poradí, v ktorom ich v prvej fáze tvoj program kreslil.

Constraints

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ pre každé volanie tvojej funkcie `BuildPattern`.

Note

Existujú algoritmy, ktoré vedia kresliť vzory tak, že naozaj vieme od seba rozlíšiť $67\,000\,000$ rôznych kariet.

Scoring

- Subtask 1 (2 points): $n \leq 2$.
- Subtask 2 (9 points): $n \leq 25$.
- Subtask 3 (15 points): $n \leq 1\,000$ a je zaručené, že medzi fázami 1 a 2 grader **neotočí žiadnu kartu**. (Môže však robiť úpravy zvyšných dvoch typov.)
- Subtask 4 (3 points): $n \leq 16\,000\,000$ a je zaručené, že medzi fázami 1 a 2 grader **neotočí žiadnu kartu**. (Môže však robiť úpravy zvyšných dvoch typov.)
- Subtask 5 (24 points): $n \leq 16\,000\,000$.
- Subtask 6 (18 points): $n \leq 40\,000\,000$.
- Subtask 7 (29 points): Bez ďalších obmedzení.

Sample Interaction

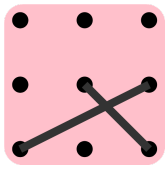
Udalosť / volanie funkcie	Návratová hodnota funkcie	Vysvetlenie
Začiatok prvej fázy.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Dostali sme kartu, na ktorej je číslo 3. Chceme na ňu nakresliť nejaký vzor. Vybrali sme si taký, ktorý tvoria dve platné úsečky: - medzi $(0, 0)$ a $(2, 1)$, - medzi $(1, 1)$ a $(2, 0)$.

`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Ako druhú sme dostali na pokreslenie kartu s číslom 1. Vybrali sme si vzor tvorený jednou platnou úsečkou: - medzi $(0,1)$ a $(0,0)$.
Koniec prvej fázy.	-	-
Začiatok druhej fázy.	-	-
`GetCardNumber(` `{{{0, 0}, {0, 1}}})`	`1`	Dostali sme vzor tvorený jednou platnou úsečkou. Tá vedie medzi $(0,0)$ a $(0,1)$. Tento vzor vyzerá rovnako ako vzor, ktorý obsahuje platnú úsečku medzi $(0,1)$ a $(0,0)$. Ide teda zjavne o vzor, ktorý sme nakreslili na kartu s číslom 1, a teda naša funkcia má vrátiť 1.

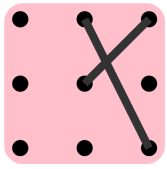
`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Tento vzor tvoria dve úsečky:
		- medzi $(1,1)$ a $(2,2)$,
		- medzi $(1,2)$ a $(2,0)$.
		Toto je vzor, ktorý vznikne zo vzoru na karte s číslom 3 otočením o 90 stupňov proti smeru ručičiek. Vrátime teda číslo 3.

Koniec druhej fázy.	-	-

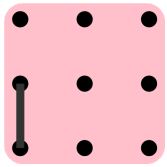
Prvý obrázok: Vzor nakreslený pri prvom volaní `BuildPattern`:



Druhý obrázok: Vzor odovzdaný ako parameter pri druhom volaní `GetCardNumber`. Všimni si, že ide o predchádzajúci vzor otočený o 90 stupňov doľava.



Tretí obrázok: Vzor nakreslený pri druhom volaní `BuildPattern` a zároveň vzor odovzdaný ako parameter pri prvom volaní `GetCardNumber`.



Sample Grader

V súbore `Cheat.zip`, ktorý dostanete, je ukážkový grader: `grader.cpp`

Ten najskôr prečíta zo štandardného vstupu číslo q. Potom q-krát zopakuje nasledovnú postupnosť krokov:

- Prečíta číslo n zo štandardného vstupu.
- Zavolá `BuildPattern(n)` a výstup si uloží do premennej p.
- Zavolá `GetCardNumber(p)` a vypíše návratovú hodnotu tejto funkcie na štandardný výstup.

Pripomíname, že tento ukážkový grader si môžeš upraviť, ak chceš.

Skompilovať ho spolu so svojim riešením vieš v termináli nasledovne:

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

V tomto príkaze `solution.cpp` je názov súboru s implementáciou tvojich funkcií (t.j. toho súboru, ktorý odovzdávaš do CMS).

Skompilovaný program vieš spusiť na danom vstupe príkazom:

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

Upozorňujeme, že tento ukážkový grader funguje ináč od skutočného. Skutočný grader sa správa tak ako bolo popísané vyššie: najskôr pri jednom samostatnom spustení skompilovaného programu nechá tvoje riešenie nakresliť všetky vzory a potom pri druhom samostatnom spustení skontroluje, či tvoje riešenie všetkým vzorom naspäť priradí správne čísla.