Chika Wants to Cheat

Feladat neve	Cheat
Bemenet	interaktív feladat
Kimenet	interaktív feladat
Időkorlát	2 másodperc
Memóriakorlát	512 MB

Chikának egy pakliban q játékkártyája van, amelyeken különböző pozitív egész számok vannak. A Shuchi'in Akadémia diáktanácsában lévő barátaival szeretne kártyázni. Nyerni akar, ezért úgy dönt, hogy titokban megjelöli a pakliban lévő kártyák hátulját.

A kártyák mindegyike négyzet alakú és 2×2 nagyságú, ahol a bal alsó sarkának a koordinátái $(0, 0)$, míg a jobb felső saroké $(2, 2)$. Chika minden kártya hátuljára rajzol egy mintát úgy, hogy később a mintát megnézve tudja, hogy melyik szám van a kártya elejére írva. A mintát a következő eljárással rajzolja: annyiszor, ahányszor csak akarja (esetleg 0-szor), kiválaszt két különböző pontot a kártyán: A-t és B-t, amelyek a kártya bal alsó sarkához viszonyítva egész koordinátájú pontok, majd egyenes **szakaszt** rajzol közöttük.

Chika csak **érvényes** szakaszokat rajzol, azaz szakasz csak két olyan A és B pont között lehet, amelyekhez nem tartozik A-tól és B-től különböző, egész koordinátájú C pont. Például a $(0, 0)$ és $(2, 2)$ közötti szakasz **nem érvényes**, mivel tartalmazza az $(1, 1)$ pontot is. A $(0, 0)$ és $(1, 1)$, valamint a $(1, 1)$ és $(2, 2)$ közötti szakaszok mindegyike **érvényes**, sőt, Chika akár mindkettőt berajzolhatja ugyanabban a mintába. A szakaszoknak nincs irányuk: az A-ból B-be rajzolt szakasz **azonos** önmagával és a fordított irányban, a B-ből az A-ba rajzolt szakasszal is.

Chika biztos akar lenni abban, hogy - függetlenül attól, hogy hogyan forgatjuk őket - felismeri a kártyákat. Egy kártyát el lehet forgatni az eredeti tájolásához képest 0, 90, 180 vagy 270 fokkal (az óramutató járásával ellentétes irányban).

A feladatod az, hogy segíts Chikának megtervezni a paklijában lévő q kártya mintáját, majd később felismerni ezeket a kártyákat.

Megvalósítás

*Ez egy két fázisból álló interaktív feladat, **mindegyik fázisban a program külön futtatásával**.*

Két függvényt kell megvalósítanod:

- A `BuildPattern` függvényt, amely visszaadja az adott kártya hátoldalára rajzolandó mintát. Ezt a függvényt az első fázisban q alkalommal hívjuk meg.
- A `GetCardNumber` függvényt, amely visszaadja a(z esetleg elforgatott) kártya elejére írt számot, amelyen egy adott, az első fázisban megrajzolt minta van. Ezt a függvényt a második fázisban q-szor hívjuk meg.

Az első függvény

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

egyetlen paramétere n: az a szám, amely a kártya elejére van írva. Vissza kell adnia egy `std::vector`-t, amely tartalmazza azokat a szakaszokat, amelyeket Chika mintaként rajzol a kártya hátoldalára, hogy később felismerhesse azt. Egy szakaszt a pontok párjaiként (`std::pair`) ábrázolunk. Egy pontot a kártya bal alsó sarkához viszonyított egész számú koordináták párjaiként (`std::pair`), (x, y)-ként adjuk meg, ahol $0 \leq x, y \leq 2$. A Chika által rajzolt összes szakasznak érvényesnek kell lennie, és páronként nem lehetnek azonosak.

Miután megadtad a mintákat a q darab kártyához, az értékelő a következő műveletek bármelyikét - minden egyes mintán tetszőleges számú alkalommal - elvégezheti:

- Elforgatja az egész mintát 0, 90, 180 vagy 270 fokkal, az óramutató járásával ellentétes irányban.
- Módosítja a szakaszok sorrendjét a minta `std::vector` ábrázolásában.
- Megváltoztatja a szakasz végpontjainak sorrendjét a mintában. (Egy A-ból B-be húzott szakaszból lehet, hogy a vele azonos, B-ből A-ba húzott szakasz lesz.)

A második függvény

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

egyetlen bemenő paramétere p, amely `std::vector` formában egy szakasz leírása, amely a Chika által a kártya hátoldalára rajzolt mintát írja le, a `BuildPattern` függvény korábbi hívásának visszatérési értéke alapján. A függvénynek a kártya elejére írt n számot kell visszaadnia. Megjegyzés: a p minta nem feltétlenül a `BuildPattern` által generált eredeti formában van. Lehetséges, hogy a fent említett három művelettel már módosítottak rajta. Az is előfordulhat, hogy a kártyák sorrendje eltér az első fázisban megadott sorrendtől.

Korlátok

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ a `BuildPattern` függvény minden egyes hívásakor.
- Megjegyzés: létezik olyan algoritmus, amellyel olyan mintákat lehet rajzolni, amivel $67\,000\,000$ különböző kártya felismerhető.

Pontozás

1. részfeladat (2 pont): $n \leq 2$.
2. részfeladat (9 pont): $n \leq 25$.
3. részfeladat (15 pont): $n \leq 1\,000$ és az értékelő **nem forgatja** el a kártyákat az első és a második fázis között (de a másik két műveletet végrehajt**hat**ja).
4. részfeladat (3 pont): $n \leq 16\,000\,000$ és az értékelő **nem forgatja** el a kártyákat az első és a második fázis között (de a másik két műveletet végrehajt**hat**ja).
5. részfeladat (24 pont): $n \leq 16\,000\,000$.
6. részfeladat (18 pont): $n \leq 40\,000\,000$.
7. részfeladat (29 pont): Nincs további megkötés.

Mintainterakció

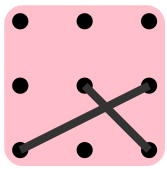
Függvényhívás	Visszatérési érték	Magyarázat
Első fázis kezdete	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Mintát kell készíteni a 3 kódolására a 2×2 méretű kártyán. 2 szakaszt rajzolunk: - a $(0,0)$ és a $(2,1)$ között, - az $(1,1)$ és a $(2,0)$ között.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Mintát kell készíteni a 1 kódolására a 2×2 méretű kártyán. 1 szakaszt rajzolunk: - a $(0,1)$ és a $(0,0)$ között.
Első fázis vége	-	-
Második fázis kezdete	-	-
`GetCardNumber({{{0, 0}, {0, 1}}})`	`1`	1 szakaszból álló minta a bemenet: - a $(0,0)$ és $(0,1)$. Ez ugyanaz a minta, mint a $(0,1)$ és $(0,0)$ pontok közötti szakasz, ami pontosan ugyanaz a minta és ugyanaz az állás (0 fokkal elforgatva), mint amit a `BuildPattern` függvény

		második hívása generált. Emiatt a visszatérési érték az 1.

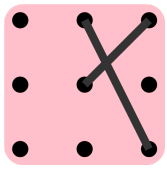
`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	2 szakaszból álló mintát kaptunk:
		- $(1, 1)$ és $(2, 2)$ közötti szakaszt,
		- $(1, 2)$ és $(2, 0)$ közötti szakaszt.
		Ez a minta, amit a `BuildPattern` függvény első hívása által generált. 90 fokkal az óramutató járásával egyezően elforgatva. Emiatt a visszatérési érték a 3.

Második fázis vége	-	-

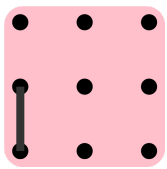
- A következő három kép, sorrendben:
- A `BuildPattern` első hívása által visszaküldött minta:



A `GetCardNumber` második hívásakor paraméterként kapott minta, amely az első minta az óramutató járásával ellentétes irányban történő 90 fokos elforgatása után:



A `BuildPattern` második hívásakor kimenetként visszaküldött minta, amely ugyanaz a minta, mint amit a `GetCardNumber` első hívás paramétereként kapott.



Mintaértékelő

A mintaértékelő, `grader.cpp`, a feladat `Cheat.zip` csatolmányában van. Beolvassa a q egész számot a standard bemenetről, majd a következő lépéseket q alkalommal végzi el:

- Beolvas egy n egész számot a standard bemenetről.
- Meghívja a `BuildPattern(n)` függvényt és a visszatérési értéket egy p változóban tárolja.
- Meghívja a `GetCardNumber(p)` függvényt és a visszatérési értéket kiírja a standard kimenetre.

A mintaértékelőt lokálisan módosíthatod.

A mintaértékelőt a megoldásoddal együtt a következő paranccsal fordíthatod le a parancssorban:

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

ahol a `solution.cpp` megoldásod, amit a CMS-be töltesz fel. Ha a programodat a mellékletben adott mintabemenettel akarod futtatni, akkor a következő paranccsal teheted meg a terminálban:

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

Megjegyzés: a mintaértékelővel ellentétben a CMS értékelője az első és a második fázisban a programot külön-külön futtatja.