ჩიკას სურს მოიტყუოს

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

ჩიკას აქვს კარტების შეკვრა, რომელშიც q რაოდენობის კარტია და ისინი გადანომრილია ურთიერთგანსხვავებული მთელი დადებითი რიცხვებით. მას სურს ამ კარტებით სხვადასხვა თამაშები ითამაშოს თავის მეგობრებთან შუჩინის აკადემიის სტუდენტური საბჭოდან, მაგრამ მას ამასთანავე ძალიან სურს მოგება და გადაწყვიტა კარტები უკანა მხრიდან ჩუმად მონიშნოს.

ყველა კარტს აქვს კვადრატული ფორმა ზომებით 2×2, სადაც ქვედა მარცხენა კუთხეს აქვს კოორდინატები $(0, 0)$, ხოლო ზედა მარჯვენა კუთხეს კი კოორდინატები $(2, 2)$. თითოეული კარტის უკანა მხარეს ჩიკა ხატავს გარკვეულ შაბლონს, რომ მერე ამ შაბლონების მიხედვით მიხვდეს, თუ რა რიცხვი აწერია შესაბამის კარტს წინა მხრიდან. ის ამ შაბლონს ხატავს შემდეგი პროცედურის გამოყენებით: იმდენჯერ, რამდენჯერაც მას სურს (შეიძლება 0-ჯერაც), ის ირჩევს ორ განსხვავებულ A და B წერტილს, რომელთაც მთელი კოორდინატები გააჩნიათ კარტის ქვედა მარცხენა კუთხის მიმართ და მათ შორის ავლებს **მონაკვეთს**.

ჩიკა არასოდეს ავლებს **ვარგის** მონაკვეთებს. ანუ მონაკვეთს ორ A და B წერტილს შორის, რომლისთვისაც არსებობს ამავე მონაკვეთზე მდებარე მთელ კოორდინატებიანი რაიმე C წერტილი (განსხვავებული A და B წერტილებისაგან). მაგალითად, მონაკვეთი $(0, 0)$ და $(2, 2)$ წერტილებს შორის არ არის **ვარგისი**, რადგან ის შეიცავს წერტილს კოორდინატებით $(1, 1)$, მაგრამ მონაკვეთები $(0, 0)$ და $(1, 1)$, ასევე $(1, 1)$ და $(2, 2)$ წერტილებს შორის ორივე **ვარგისია** და ჩიკას ორივე მათგანის გავლებაც კი შეუძლია ერთი და იგივე შაბლონის დახატვისას. ასევე შევნიშნოთ, რომ მონაკვეთებს მიმართულება არ გააჩნიათ: მონაკვეთი გავლებული A-დან B-მდე **იდენტურია** როგორც თავისთავის, ასევე B-დან A-მდე გავლებული მონკვეთისა.

ჩიკას სურს დარწმუნებული იყოს, რომ ის ამოიცნობს თავის კარტებს მიუხედავად იმისა, როგორადაც არ უნდა იყვნენ ისინი მობრუნებული. კარტის მობრუნება შესაძლებელია 0, 90, 180 ან 270 გრადუსით საათის ისრის მოძრაობის საწინააღმდეგო მიმართულებით მისი საწყისი მდგომარეობის მიმართ.

თქვენი ამოცანაა დაეხმაროთ ჩიკას შექმნას შაბლონები შეკვრაში არსებული ყველა q რაოდენობის კარტისათვის ისე, რომ შემდეგ ამ კარტების ამოცნობა შეძლოს.

იმპლემენტაცია

*ეს არის ორეტაპიანი ინტერაქტიული ამოცანა, **თითოეული ეტაპი მოიცავს თქვენი პროგრამის ცალკე გაშვებას**.* თქვენ უნდა მოახდინოთ შემდეგი ორი ფუნქციის იმპლემენტაცია:

- A `BuildPattern` ფუნქცია, რომელიც აბრუნებს მოცემული კარტის უკანა მხარეს დასახატ შაბლონს. ეს ფუნქცია პირველ ეტაპზე გამოძახებული იქნება q-ჯერ.
- A `GetCardNumber` ფუნქცია, რომელიც აბრუნებს პირველ ეტაპზე დახატული, მოცემული შაბლონის შემცველი კარტის (შესაძლებელია მობრუნებულის) ნომერს. ეს ფუნქცია მეორე ეტაპზე გამოძახებული იქნება q-ჯერ.

პირველი ფუნქცია

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

ღებულობს ერთადერთ n პარამეტრს - რიცხვს, რომელიც კარტის წინა მხარესაა დაწერილი. თქვენ უნდა დააბრუნოთ `std::vector`, რომელიც შეიცავს ჩიკას მიერ კარტის უკანა გვერდზე მისი შემდგომი გამოცნობის მიზნით გავლებულ მონაკვეთებს. მონაკვეთი წარმოდგენილია როგორც წერტილების წყვილი `std::pair`, ხოლო წერტილი წარმოდგენილია როგორც მთელ კოორდინატთა (კარტის ქვედა მარცხენა კუთხის მიმართ) წყვილი `std::pair` (x, y), სადაც $0 \leq x, y \leq 2$. ყველა მონაკვეთი, რომელთაც ჩიკა ავლებს, უნდა იყოს ვარგისი და წყვილ-წყვილად არაიდენტური. გარანტირებულია, რომ BuildPattern-ის ყოველი q გამოძახება n პარამეტრის განსხვავებულ მნიშვნელობას მიიღებს.

მიიღებს რა ყველა შაბლონს q რაოდენობის კარტისათვის, გრადერს თითოეულ შაბლონზე შეუძლია შეასრულოს ნებისმიერი შემდეგი ოპერაციებიდან ნებისმიერ რაოდენობაჯერ:

- მოაბრუნოს მთლიანი შაბლონი 0, 90, 180 or 270 გრადუსით საათის ისრის მოძრაობის საწინააღმდეგო მიმართულებით.
- შეცვალოს მონაკვეთების მიმდევრობა შაბლონის `std::vector` წარმოდგენაში.
- შეცვალოს შაბლონში მონაკვეთის ბოლო წერტილების მიმდევრობა. (მონაკვეთიA-დან B-მდე შეიძლება გახდეს იდენტური მონაკვეთისა B-დან A-მდე.)

მეორე ფუნქცია,

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

ღებულობს ერთადერთ p პარამეტრს, მონაკვეთების `std::vector`-ს, რომლებიც აღწერენ ჩიკას მიერ კარტის უკანა მხარეს, უფრო ადრე გამოძახებული თქვენი `BuildPattern`-ის მიერ დაბრუნებული მნიშვნელობის მიხედვით დახატულ შაბლონს. ფუნქციამ უნდა დააბრუნოს რიცხვი n, რომელიც კარტის წინა მხარესაა დაწერილი. შეგახსენებთ, რომ ფიგურა p აუცილებლად არ იმყოფება `BuildPattern`-ის მიერ დაბრუნებულ საწყის ფორმაში. მასზე შეიძლებოდა ზემოთ

აღწერილი სამი ოპერაცია შესრულებულიყო. ასევე შესაძლებელია კარტების მიმდევრობა განსხვავდებოდეს იმ მიმდევრობისაგან, როგორც ისინი პირველ ეტაპზე იყვნენ მოცემული, მაგრამ გარანტირებულია, რომ თითოეული კარტი გამოყენებული იქნება ზუსტად ერთხელ

შეზღუდვები

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ ფუნქცია `BuildPattern`-ის ყველა გამოძახებისათვის.
- შევნიშნოთ, რომ არსებობენ შაბლონების აგების (დახატვის) ალგორითმები, რომლებსაც შეუძლიათ $67\,000\,000$ განსხვავებული კარტის ამოცნობა.

შეფასება

- ქვეამოცანა 1 (2 ქულა): $n \leq 2$.
- ქვეამოცანა 2 (9 ქულა): $n \leq 25$.
- ქვეამოცანა 3 (15 ქულა): $n \leq 1\,000$ და გრადერი **არ მოაბრუნებს** შაბლონებს 1-ლ და მე-2 ეტაპებს შორის. (გრადერს **შეუძლია** შეასრულოს დანარჩენი ორი ოპერაცია.)
- ქვეამოცანა 4 (3 ქულა): $n \leq 16\,000\,000$ და გრადერი **არ მოაბრუნებს** შაბლონებს 1-ლ და მე-2 ეტაპებს შორის. (გრადერს **შეუძლია** შეასრულოს დანარჩენი ორი ოპერაცია.)
- ქვეამოცანა 5 (24 ქულა): $n \leq 16\,000\,000$.
- ქვეამოცანა 6 (18 ქულა): $n \leq 40\,000\,000$.
- ქვეამოცანა 7 (29 ქულა): დამატებითი შეზღუდვების გარეშე.

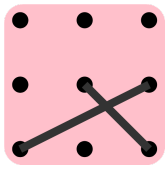
ინტერაქციის ნიმუში

ფუნქციის გამოძახება	დაბრუნებული მნიშვნელობა	განმარტება
პირველი ეტაპი იწყება.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	ჩვენ უნდა შევქმნათ შაბლონი რიცხვი 3-სათვის 2×2 ზომის კარტზე. ჩვენ გადავწყვიტეთ გავავლოთ 2 მონაკვეთი: - $(0, 0)$-სა და $(2, 1)$-ს შორის, - $(1, 1)$-სა და $(2, 0)$-ს შორის.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	ჩვენ უნდა შევქმნათ შაბლონი რიცხვი 1-სათვის 2×2 ზომის კარტზე. ჩვენ გადავწყვიტეთ გავავლოთ 1 მონაკვეთი: - $(0, 1)$-სა და $(0, 0)$-ს შორის.
პირველი ეტაპი მთავრდება.	-	-
მეორე ეტაპი იწყება.	-	-

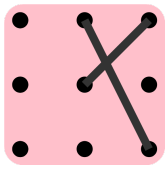
`GetCardNumber( {{{0, 0}, {0, 1}}})`	1	ვღებულობთ ერთი მონაკვეთისაგან შედგენილ შაბლონს: - $(0, 0)$-სა და $(0, 1)$-ს შორის. ეს იგივე შაბლონია, რომლის მიღებაც ჩვენ შეგვეძლო შემდეგი მონაკვეთის გავლებით: - $(0, 1)$-სა და $(0, 0)$-ს შორის. ეს ზუსტად იგივე შაბლონია იგივე ორიენტაციით (მობრუნებულია 0 გრადუსით), რომელიც ჩვენ დავაბრუნეთ `BuildPattern` ფუნქციის მეორე გამოძახებით. ამიტომ ჩვენ ვაბრუნებთ 1-ს.
`GetCardNumber( {{{1, 1}, {2, 2}}, {{1, 2}, {2, 0}}})`	3	ვღებულობთ ორი მონაკვეთისაგან შედგენილ შაბლონს: - $(1, 1)$-სა და $(2, 2)$-ს შორის, - $(1, 2)$-სა და $(2, 0)$-ს შორის. ეს ის შაბლონია, რომელიც ჩვენ დავაბრუნეთ `BuildPattern` ფუნქციის პირველი გამოძახებით, მობრუნებული 90 გრადუსით საათის ისრის მოძრაობის საწინააღმდეგო მიმართულებით. ამიტომ ჩვენ ვაბრუნებთ 3-ს.
მეორე ეტაპი მთავრდება.	-	-

შემდეგი სამი ნახაზი მიმდევრობით წარმოადგენენ:

- გამოტანის სახით დაბრუნებულ შაბლონს `BuildPattern` ფუნქციის პირველი გამოძახებისას:

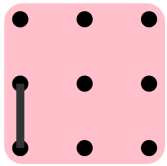


- `GetCardNumber` ფუნქციის მეორე გამოძახებისას პარამეტრის სახით მიღებულ შაბლონს, რომელიც წარმოადგენს პირველ შაბლონს მისი 90 გრადუსით მობრუნებისას საათის ისრის მოძრაობის საწინააღმდეგო მიმართულებით.



- გამოტანის სახით დაბრუნებულ შაბლონს `BuildPattern` ფუნქციის მეორე გამოძახებისას, რომელიც ასევე წარმოადგენს `GetCardNumber` ფუნქციის პირველი გამოძახებისას

არგუმენტის სახით დაბრუნებულ შაბლონს.



სანიმუშო გრადერი

დავალების `Cheat.zip` დანართში წარმოდგენილი სანიმუშო `grader.cpp` გრადერი სტანდარტული შეტანიდან კითხულობს მთელ q რიცხვს, ხოლო შემდეგ q-ჯერ ასრულებს შემდეგ ბიჯებს:

- კითხულობს მთელ n რიცხვს სტანდარტული შეტანიდან;
- იძახებს `BuildPattern(n)`-ს და დაბრუნებულ მნიშვნელობას ინახავს p ცვლადში.
- იძახებს `GetCardNumber(p)`-ს და ბეჭდავს დაბრუნებულ მნიშვნელობას სტანდარტულ გამოტანაში.

თქვენ შეგიძლიათ შეცვალოთ თქვენი გრადერი ლოკალურად, თუ ამისი სურვილი გაქვთ.

თქვენს ამონახსნთან ერთად სანიმუშო გრადერის კომპილაციისათვის თქვენ შეგიძლიათ გამოიყენოთ შემდეგი ბრძანებები ტერმინალის სტრიქონში:

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

სადაც `solution.cpp` არის CMS-ში გასაგზავნი თქვენი ამონახსნის ფაილი. მიბმულ ფაილში წარმოდგენილი შესატანი მონაცემების ნიმუშით პროგრამის გასაშვებად ტერმინალის სტრიქონში შეიტანეთ შემდეგი ბრძანება: :

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

**ყურადღება მიაქციეთ იმას, რომ სანიმუშო გრადერისაგან განსხვავებით, ნამდვილი გრადერი პირველ ეტაპს და მეორე ეტაპს CMS-ში თქვენი პროგრამის ცალ-ცალკე გაშვებისას შეასრულებს.