# Ања жели да вара

| Назив проблема | Варање |
|---|---|
| Улаз | Интерактивни задатак |
| Излаз | Интерактивни задатак |
| Временско ограничење | 2 секунде |
| Меморијско ограничењењ | 512 мегабајта |

Ања има шпил са $q$ карата за игру нумерисаних различитим позитивним целим бројевима. Она жели да игра неке игре са својим пријатељима из МатГим-а користећи те карте, али такође жели да победи, па је одлучила да тајно означи (маркира) задње стране карата из свог шпила.

Све карте имају облик квадрата димензија $2 \times 2$, тако да леви доњи угао има координате $(0, 0)$, а горњи десни угао има координате $(2, 2)$. Ања црта одређену шару (слику) на позадини сваке карте, тако да касније на основу слике на позадини карте зна који је број написан на предњој страни карте. Она црта слику користећи следећу процедуру: Онолико пута колико жели (може бити и $0$ пута), она фиксира две различите тачке $A$ и $B$ које имају целобројне координате у односу на доњи леви угао карте и црта **дуж (праву линију)** између тих тачака.

Међутим, Ања може цртати само **исправне** дужи, а дуж која спаја неке две тачке $A$ и $B$ је исправна ако не садржи ниједну тачку $C$ (различиту од тачака $A$ и $B$) са целобројним координтама. На пример, дуж која спаја тачке $(0, 0)$ и $(2, 2)$ је **није исправна** јер садржи тачку $(1, 1)$, али су дужи између $(0, 0)$ и $(1, 1)$, односно између $(1, 1)$ и $(2, 2)$ **исправне**, и Ања може нацртати чак и обе на истој слици. Такође, приметите да дужи немају оријентацију (смер): Дуж од тачке $A$ до тачке $B$ је **идентична** са собом и са дужи нацртаном у супротном смеру, од тачке $B$ до тачке $A$.

Такође је врло важно да Ања сигурно може да препозна карту без обзира како је она ротирана. Карта може бити ротирана за $0$, $90$, $180$ или $270$ степени у смеру супротном од смера кретања казаљки на часовнику у односу на оригиналну оријентацију.

Ваш задатак је да помогнете Ањи да осмисли слике за датих $q$ карата у њеном шпилу и да касније препозна те карте.

# Имплементација

*Ово је интерактивни задатак са две фазе,* ***свака фаза захтева засебно извршавање вашег програма***. Ви треба да имплементирате две функције:

- Функција `BuildPattern` враћа слику која ће бити нацртана на позадини дате карте. Ова функција се позива $q$ пута у првој фази.
- Функција `GetCardNumber` враћа број на карти (можда ротираној) на којој се налази слика нацртана у првој фази. Ова функција ће бити позвана $q$ пута у другој фази.

Прва функција

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

добија један аргумент $n$, и то је број који је написан на предњој страни карте. Ви треба да вратите резултат типа `std::vector` који садржи дужи које Ања црта као слику на позадини те карте како би касније могла да је препозна. Свака дуж је репрезентована као `std::pair` тачака, а свака тачка је репрезентована као `std::pair` $(x, y)$ целих бројева који представљају координате те тачке релативно у односу на доњи леви угао карте, при чему важи $0 \leq x, y \leq 2$. Све дужи које исцртава Ања треба да буду исправне и да буду у паровима различите. Гарантује се да ће у свих $q$ позива функције `BuildPattern` бити различите вредности аргумента $n$.

Након добијања слика за свих $q$ карата, грејдер може да уради било коју од следећих операција произвољан број пута, на свакој од слика:

- Ротира целу слику за $0$, $90$, $180$ или $270$ степени у смеру супротном од кретања казаљки часовника.
- Промени редослед дужи у `std::vector` репрезентацији слике.
- Промени редослед крајева неке дужи на слици. (Дуж од тачке $A$ до тачке $B$ прелази у њој идентичну дуж од тачке $B$ до тачке $A$.)

Друга функција,

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

добија један аргумент $p$, који представља `std::vector` дужи које описују слику коју је Ања нацртала на задњој страни неке карте, и која је базирана на вредности која је враћена у претходном позиву ваше функције `BuildPattern`. Ваша функција треба да врати број $n$ који је написан на предњој страни те карте . Сетите се да слика $p$ не мора бити у оној оригиналној форми коју је вратила функција `BuildPattern`; она може бити добијена применом три горе

описане операције на оригиналну слику нацртану на задњој страни карте. Редослед карата у другој фази се може разликовати од редоследа у првој фази.

## Ограничења

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ за све позиве функције `BuildPattern`.
- Напомињемо да постоји алгоритам који може направити слике помоћу којих се може препознати $67\,000\,000$ различитих карти.

## Бодовање

- Подзадатак 1 (2 поена): $n \leq 2$.
- Подзадатак 2 (9 поена): $n \leq 25$.
- Подзадатак 3 (15 поена): $n \leq 1\,000$ и грејдер **неће ротирати** слике између фаза 1 и 2. (Грејдер **може** извршити друге две операције.)
- Подзадатак 4 (3 поена): $n \leq 16\,000\,000$ и грејдер **неће ротирати** слике између фаза 1 и 2. (Грејдер **може** извршити друге две операције.)
- Подзадатак 5 (24 поена): $n \leq 16\,000\,000$.
- Подзадатак 6 (18 поена): $n \leq 40\,000\,000$.
- Подзадатак 7 (29 поена): Нема додатних ограничења.

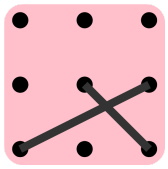
## Пример интеракције

| Позив функције | Враћена вредност | Објашњење |
|---|---|---|
| Почиње прво извршавање. | - | - |
| `BuildPattern(3)` | `{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}` | Направили смо слику за карту са бројем $3$ на $2 \times 2$ карти. Одлучили смо да нацртамо $2$ дужи:<br>- између $(0, 0)$ и $(2, 1)$,<br>- између $(1, 1)$ и $(2, 0)$. |
| `BuildPattern(1)` | `{{{0, 1}, {0, 0}}}` | Направили смо слику за карту са бројем $3$ на $2 \times 2$ карти. Одлучили смо да нацртамо $1$ дуж:<br>- између $(0, 1)$ и $(0, 0)$. |
| Завршено прво извршавање. | - | - |
| Почиње друго извршавање. | - | - |
| `GetCardNumber(` | `1` | Добили смо слику која се састоји од |

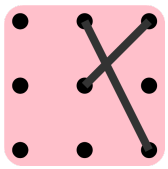
| | | |
|---|---|---|
| `{{{0, 0}, {0, 1}}})` | | 1 дужи:<br>- између $(0,0)$ и $(0,1)$.<br>Ово је слика иста као слика добијена цртањем дужи:<br>- између $(0,1)$ и $(0,0)$<br>која је у потпуности иста као слика са истом оријентацијом (ротираном за 0 степени) која је добијена другим позивом функције `BuildPattern`. Због тога враћамо резултат $1$. |
| `GetCardNumber(`<br>`{{{1, 1}, {2, 2}},`<br>`{{1, 2}, {2, 0}}})` | `3` | Добијамо слику која се састоји од 2 дужи:<br>- између $(1,1)$ и $(2,2)$,<br>- између $(1,2)$ и $(2,0)$.<br>Ово је слика добијена првим позивом функције `BuildPattern` ротирана за $90$ степени у смеру супротном од кретања казаљки часовника. Због тога враћамо резултат $3$. |
| Завршава се друго извршавање. | - | - |

Следеће три слике репрезентују (у том редоследу):

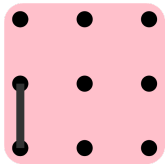
- Слику враћену као резултат при првом позиву функције `BuildPattern`:



- Слику послату као параметар за други позив функције `GetCardNumber`, која представља прву слику после ротације за 90 степени супротно од кретања казаљки часовника.



- Слику враћену при другом позиву функције to `BuildPattern`, која се поклапа са сликом која је послата као аргумент у првом позиву функције `GetCardNumber`.



## Пример грејдера

Грејдер `grader.cpp`, који се налази у прилогу `Cheat.zip`, чита цео број $q$ са стандардног улаза и након тога извршава следеће кораке $q$ пута:

- Чита цео број $n$ са стандардног улаза.
- Позива `BuildPattern(n)` и памти вредност коју је функција вратила у променљивој $p$.
- Позива функцију `GetCardNumber(p)` и исписује резултат који је вратила функција на стандардни излаз.

Ви можете модификовати ваш грејдер локално, ако желите.

Да би превели тај пример грејдера заједно са вашим решењем, користите следећу команду у терминалу:

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

Овде је `solution.cpp` ваше решење које ћете предати на CMS. Да би извршили програм са примером улаза који се налази у прилогу, куцате следећу команду у терминалу:

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

**Моломо вас да обратите пажњу, да ће за разлику од грејдера из прилога, прави грејдер на CMS извршити прву и другу фазу у посебним извршавањима вашег програма.**