千花はズルをしたい（Chika Wants to Cheat）

問題名	千花はズルをしたい
入力	(インタラクティブ問題)
出力	(インタラクティブ問題)
実行時間制限	2秒
メモリ制限	512MB

千花は表面に整数が書かれた q 枚のカードを持っている．彼女は，秀知院学園の生徒会の友達とカードゲームを行おうと思っているが，絶対に勝ちたいため，カードの裏面にこっそり模様を付けることにした．

すべてのカードは大きさ 2×2 の正方形の形をしており，左下座標は $(0, 0)$，右上座標は $(2, 2)$ である．千花は模様を見るだけで表面に書かれた整数がわかるように，カードの裏面に模様を描く．彼女は模様を以下の方法で作ろうと思っている：2 つの異なる格子点（整数座標の点）A, B を選び，2 点間をまっすぐ結ぶ **線分** を描く，という操作を 0 回以上行う．

ここで，千花は **有効な線分** しか描くことはない．ただし有効な線分とは，端点 A, B の間に格子点 C（A, B とは異なる）が存在しない線分のことを指す．たとえば座標 $(0, 0)$ と $(2, 2)$ を結ぶ線分は，座標 $(1, 1)$ を通るため有効ではない．一方，$(0, 0)$ と $(1, 1)$ を結ぶ線分や，$(1, 1)$ と $(2, 2)$ を結ぶ線分は有効であり，千花はその両方を描くことも許される．

なお，線分には向きが存在しない．端点 A から端点 B に向けて描かれた線分は，端点 B から端点 A に向けて描かれた線分と **同一のもの** である．

さて，千花はどのようにカードが回転されても認識できるようにしたいと思っている．ここで，カードは元々の状態から反時計回りに $0, 90, 180, 270$ 度回転される可能性がある．あなたの課題は，千花が模様を作り，その後 q 個のカードを識別するのを助けることである．

実装の詳細

*この問題は，ステージ 1 およびステージ 2 の 2 段階からなるインタラクティブ問題であり，**<u>各ステージは別々に実行される．</u>*** あなたは以下の 2 つの関数を実装しなければならない．

- 関数 `BuildPattern`：与えられたカードの裏面に描く模様を返す．この関数は，ステージ 1 で q 回呼び出される．
- 関数 `GetCardNumber`：ステージ 1 で描かれたカードのうち 1 つ（を適宜回転させたもの）の模様が与えられるので，そのカードに書かれた整数を返す．この関数は，ステージ 2 で q 回呼び出される．

1 つ目の関数：

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

この関数は，1 つの整数 n を引数とする．n はカードの表面に書かれた整数である．あなたは，千花がカードの裏面に描くべき線分のリストを，`std::vector` 型で返さなければならない．1 つの線分は，点を要素とする `std::pair` 型で表される．点はカードの左下隅を原点とした整数座標 (x, y) ($0 \leq x, y \leq 2$)を要素とする `std::pair` 型で表される．千花が描くすべての線分は有効なものであり，どの 2 つの線分も異なるものでなければならない．なお，q 回の呼び出しにおける n の値はすべて異なることが保証される．

q 個すべてのカードの模様を受け取った後，ジャッジシステムはそれぞれの模様に対し，以下の 3 種類の操作を何回か行うことがある．

- 模様全体を反時計回りに $0, 90, 180, 270$ 度回転させる．
- `std::vector` 型で与えられた線分の順序を変える．
- 各線分の端点の順序を入れ替える．（すなわち，端点 A から B に向けて描かれた線分は，端点 B から A に向けて描かれた線分に変わる可能性がある）

2 つ目の関数：

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

この関数は，1 つの変数 p を引数とする．p は千花がカードの裏面に描いた模様を `std::vector` 型で表したものであり，事前に呼び出された関数 `BuildPattern` の返り値に基づいている．この関数は，カードの表面に書かれた整数 n を返さなければならない．

ここで，模様 p は関数 `BuildPattern` の呼び出しで返された元の形と同じとは限らないことに注意すること（前述の 3 種類の操作を行う可能性がある）．また，ステージ 2 で聞くカードの順番は，ステージ 1 で聞くカードの順番と異なる可能性があることに注意すること．なお，各カードは 1 回ずつ聞かれることが保証される．

制約

- $1 \leq q \leq 10\,000$.
- 関数 `BuildPattern` のすべての呼び出しについて $1 \leq n \leq 67\,000\,000$.
- なお，$67\,000\,000$ 個の異なるカードを識別できるような模様を作るアルゴリズムは存在する．

小課題

1. (2 点) $n \leq 2$.
2. (9 点) $n \leq 25$.
3. (15 点) $n \leq 1\,000$．ジャッジシステムは，ステージ 1 と 2 の間に**模様を回転させることはない．（他の 2 種類の操作を行う可能性はある**ことに注意すること）
4. (3 点) $n \leq 16\,000\,000$．ジャッジシステムは，ステージ 1 と 2 の間に**模様を回転させることはない．（他の 2 種類の操作を行う可能性はある**ことに注意すること）
5. (24 点) $n \leq 16\,000\,000$.
6. (18 点) $n \leq 40\,000\,000$.
7. (29 点) 追加の制約はない．

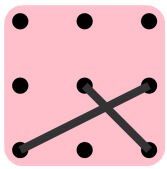
例

関数の呼び出し	返り値	説明
(一回目の実行が始まる)	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	あなたは，整数 3 が書かれた 2×2 のカードに模様を描かなければならない．あなたは以下の 2 つの線分を描くことに決めた： - $(0,0)$ と $(2,1)$ を結ぶ． - $(1,1)$ と $(2,0)$ を結ぶ．
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	あなたは，整数 1 が書かれた 2×2 のカードに模様を描かなければならない．あなたは以下の 1 つの線分を描くことに決めた： - $(0,1)$ と $(0,0)$ を結ぶ．
(一回目の実行が終わる)	-	-
(二回目の実行が始まる)	-	-
`GetCardNumber({{{0, 0}, {0, 1}}})`	`1`	引数は以下の 1 つの線分からなる模様である： - $(0,0)$ と $(0,1)$ を結ぶ． この模様は，以下の線分を描くことで作られる模様と同一のものである： - $(0,1)$ と $(0,0)$ を結ぶ．

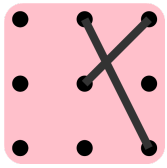
		なお，これは `BuildPattern` の 2 回目の呼び出しで作られた模様を，状態を変えずにそのまま残したもの（0 度回転させたもの）と全く同じである．したがって，1 と返さなければならない．
`GetCardNumber({{{1, 1}, {2, 2}}, {{1, 2}, {2, 0}}})`	3	引数は以下の 2 つの線分からなる模様である： - $(1,1)$ と $(2,2)$ を結ぶ． - $(1,2)$ と $(2,0)$ を結ぶ． これは，`BuildPattern` の 1 回目の呼び出しで作られた模様を，反時計回りに 90 度回転させたものである．したがって，3 と返さなければならない．
(二回目の実行が終わる)	-	-

以下の 3 つの画像は，順に次のようなものになっている：

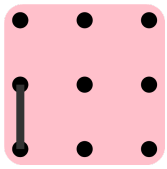
- 関数 `BuildPattern` の 1 回目の呼び出しで返された模様．



- 関数 `GetCardNumber` の 2 回目の呼び出しの引数に対応した模様．これは，上から 1 つ目の画像を反時計回りに 90 度回転させたものである．



- 関数 `BuildPattern` の 2 回目の呼び出しで返された模様．これは，関数 `GetCardNumber` の 1 回目の呼び出しの引数に対応した模様と同一のものである．



採点プログラムのサンプル（Sample Grader）

配布された `Cheat.zip` の中に入っている，採点プログラムのサンプル `grader.cpp` は，整数 q を標準入力から受け取った後，以下の処理を q 回行う：

- 整数 n を標準入力から受け取る．
- 関数 `BuildPattern(n)` を呼び出し，返り値を変数 p に保存する．
- 関数 `GetCardNumber(p)` を呼び出し，返り値を標準出力に出力する．

もし必要であれば，あなたは採点プログラムのサンプルを勝手に書き換えても良い．

採点プログラムのサンプルと，あなたの解答プログラムをコンパイルする方法の一つとして，Terminal 上で以下のコマンドを打つという手がある．

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

ここで `solution.cpp` は，CMS 上で提出されるべき解答プログラムである．また，配布されたファイルの中に入っている入出力例を用いてプログラムを実行する方法の一つとして，Terminal 上で以下のコマンドを打つという手がある．

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

注意：採点プログラムのサンプルとは異なり，CMS 上の実際のジャッジシステムでは，ステージ 1 とステージ 2 を別々に実行する，つまりプログラムは二度実行されることに注意せよ．