Kvadrat

U 4x4 kvadrat nekim redom su upisani brojevi 1, 2, ..., 16. Napravite klasu Kvadrat na kojoj su definirane sljedeće funkcije i operatori (vidi i donji primjer):

- Konstruktor, koji ne prima ništa, te inicijalizira Kvadrat ovako:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

- Operator >, koji djeluje ovako: ako je K kvadrat, a p=(r, n) podatak tipa pair<int, int>, onda operacija K > p mijenja kvadrat K tako da se redak r „zarotira“ udesno za n mjesta. Možete pretpostaviti da su brojevi r i n iz skupa {1, 2, 3, 4}. Omogućite ulančavanje. **Ovaj operator obavezno implementirajte kao virtualni član klase Kvadrat**!
- Operator ^, koji djeluje ovako: ako je K kvadrat, a p=(s, n) podatak tipa pair<int, int>, onda operacija K ^ p mijenja kvadrat K tako da se stupac s „zarotira“ prema gore za n mjesta. Možete pretpostaviti da su brojevi s i n iz skupa {1, 2, 3, 4}. Omogućite ulančavanje. **Ovaj operator obavezno implementirajte kao virtualni član klase Kvadrat**!
- Operator (), koji djeluje ovako: ako je K kvadrat, onda za int-ove r i s operacija K(r, s) vraća broj koji se nalazi u retku r i stupcu s. Korištenjem ovog operatora, moguće je i promijeniti broj koji piše na nekom polju, npr. K(4, 2)=6. Uočite da je moguće da nakon ovog poziva u neka dva polja pišu isti brojevi. Možete pretpostaviti da će svi ostali operatori biti pozivani samo nad kvadratima u kojima su svi brojevi međusobno različiti. Također, možete pretpostaviti da će r, s uvijek biti iz skupa {1, 2, 3, 4}.
- Operator ==, koji djeluje ovako: za kvadrate K i L, operacija K == L vraća int koji predstavlja broj polja na kojima u kvadratu K i kvadratu L pišu isti brojevi.
- Operator [], koji djeluje ovako: za string S i kvadrat K, operacija K[S] vraća uređeni par (r, s) cijelih brojeva takvih da se broj zapisan u stringu S nalazi u r-tom retku i s-tom stupcu od K. Ako u stringu S nije zapisan broj iz skupa {1, 2, ..., 16}, onda ovaj operator treba vratiti uređeni par (0, 0).
- Prefix operator++ koji polja na vanjskom rubu kvadrata rotira za jedno mjesto u smjeru kazaljke na satu, te prefix operator -- koji polja na vanjskom rubu kvadrata rotira za jedno mjesto u smjeru suprotnom od kazaljke na satu. Implementirajte i odgovarajuće postfix operatore koji na isti način rotiraju središnja 4 polja u kvadratu. Ne treba omogućiti ulančavanje.
- Operator !, koji djeluje ovako: ako je K kvadrat, onda operacija !K nizom poziva operatora > i ^ dovodi K u inicijalno stanje. Pri tome operatore > i ^ iz operatora ! smijete ukupno pozvati maksimalno 1000 puta. (Uočite da unutar funkcije koja definira operator ! smijete deklarirati i druge varijable tipa Kvadrat, te na njima po volji mnogo puta pozivati bilo koje operacije. U ograničenje od 1000 poziva ulaze samo pozivi > i ^ na varijabli K. Vidi primjer.) **Ovaj operator mora biti član klase Kvadrat.**

Napomene:

- Operatore (osim !, > i ^) smijete definirati bilo kao članove, bilo kao friend funkcije klase Kvadrat.
- Funkcije i operatore označite sa const svugdje gdje je to primjereno.
- Operator ! će se pozivati samo u 2 testna primjera. Pri tome, u jednom od njih će kvadrate biti moguće svesti na inicijalni pomoću **samo 4 ili manje** poziva operatora > i ^. U oba testna primjera, operator ! će biti pozvan za 3 različita kvadrata.
- Operatore morate implementirati tako da bude omogućeno brojanje broja poziva, kao u primjeru dolje (ako ih implementirate „kao na vježbama“, to će biti ispunjeno).
- Vaš program smije koristiti maksimalno 512MB memorije.
- Smijete definirati i druge pomoćne funkcije i operatore.

Primjer datoteke main.cpp:

```
#include <iostream>
#include "kvadrat.h"

using namespace std;

class TestKvadrat : public Kvadrat
{
public:
    int brojPoziva;
    TestKvadrat &operator>( const pair<int, int> &P )
    {
        ++brojPoziva;
        Kvadrat::operator>( P );
        return *this;
    }
};


int main( void )
{
    TestKvadrat K;

    K > make_pair( 3, 1 ); // 3. red rotiramo udesno za 1
    //  1  2  3  4
    //  5  6  7  8
    // 12  9 10 11
    // 13 14 15 16

    K ^ make_pair( 4, 2 ); // 4. red rotiramo gore za 2
    //  1  2  3 11
    //  5  6  7 16
    // 12  9 10  4
    // 13 14 15  8

    cout << K( 1, 4 ) << endl; // ispise 11

    pair<int, int> p = K[ "12" ];
    cout << p.first << ", " << p.second << endl; // ispise 3, 1

    K.brojPoziva = 0;
    !K; // poziva npr K ^ ( 4, 2 ), pa K > ( 3, 3 )
    //  1  2  3  4
    //  5  6  7  8
    //  9 10 11 12
    // 13 14 15 16
    // ovo bi trebalo uspjesno prebrojati broj poziva operatora > unutar !K:
    cout << K.brojPoziva << endl;

    Kvadrat L;
    if( ( K == L ) == 16 )
```

```
        cout << "ok" << endl; // ovo treba biti istina

    ++L;
    //  5  1  2  3
    //  9  6  7  4
    // 13 10 11  8
    // 14 15 16 12

    L--;
    //  5  1  2  3
    //  9  7 11  4
    // 13  6 10  8
    // 14 15 16 12

    L( 1, 1 ) = 15; L( 4, 2 ) = 5;
    // 15  1  2  3
    //  9  7 11  4
    // 13  6 10  8
    // 14  5 16 12

    return 0;
}
```