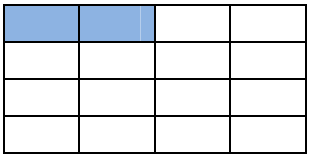
Ploca

Na NxN ploči smještena je dvodimenzionalna figurica dimenzija 2x1. Figurica je s jedne strane plave boje, a s druge crvene. Također, na ploči se na nekim mjestima mogu nalaziti kamenčići. Napravite klasu Ploca na kojoj su definirane sljedeće funkcije i operatori (vidi primjer):

- Konstruktor, koji prima prirodni broj 1 < N < 100, te inicijalizira NxN ploču tako da se na njoj ne nalazi niti jedan kamenčić, a da je figurica smještena ovako (slika je za N=4):



- Operator(), koji djeluje ovako: ako je P ploča, a x=(r, s) uređeni par cijelih brojeva, onda operacija P(x) postavlja kamenčić u r-ti redak i s-ti stupac ako se tamo već ne nalazi niti kamenčić niti figurica. Ako se tamo već nalazi kamenčić, ovaj operator ga uklanja. Ako se tamo nalazi figurica, operator ne radi ništa. Možete pretpostaviti da su r, s iz skupa {1, ..., N}. Operator vraća int, i to 0 ako se sada na mjestu (r, s) ne nalazi ništa, 1 ako se nalazi kamenčić, a 2 ako se nalazi figurica.
- Operator *, koji djeluje ovako: ako je P ploča, onda *P vraća uređeni par x=(r, s) cijelih brojeva takvih da je gornji lijevi kut figurice smješten u r-tom retku i s-tom stupcu ploče.
- Operator ~, koji djeluje ovako: ako je P ploča, onda ~P vraća string „plava“ ili „crvena“, ovisno tome koja se strana figurice trenutno vidi.
- Prefix operator ++, koji djeluje ovako: ako je P ploča, onda operacija ++P prevrne figuricu oko njezinog desnog ruba, i to samo ako se na mjestu kojeg bi figurica zauzimala nakon rotacije ne nalazi kamenčić i ako ne bi ispala s ploče (u protivnom, ne radi ništa). Slično, prefix operator -- prevrne figuricu preko njenog lijevog ruba, postfix operator ++ preko gornjeg, a postfix operator -- preko donjeg. Ne treba omogućiti ulančavanje ovih operatora. **Ove operatore obavezno implementirajte kao virtualne članove klase Ploca.**
- Operator >>, koji djeluje ovako: ako je P ploča, a phi int, onda operacija P >> phi zarotira figuricu za phi stupnjeva u smjeru kazaljke na satu, i to oko njenog gornjeg lijevog polja. Pri tome je phi iz skupa {90, 180, 270} (u protivnom operator ne radi ništa). Operator vrši rotaciju samo u slučaju kada se na ciljnom položaju figurice ne nalazi kamenčić. Omogućite ulančavanje ovog operatora. **Ovaj operator obavezno implementirajte kao virtualni član klase Ploca.**
- Operator ==, koji djeluje ovako: ako su P i Q ploče istih dimenzija, onda P==Q vraća int koji kaže na koliko postoji parova (r, s) takvih da se i na ploči P i na ploči Q u r-tom retku i s-tom stupcu nalaze kamenčići. Ako nisu iste dimenzije, vraća 0.
- Operator !, koji djeluje ovako: ako je P ploča, onda !P nizom poziva operatora ++, -- i >> dovodi figuricu u njezin inicijalni položaj (kao na gornjoj slici: položena je horizontalno, gornji lijevi kut na (1, 1), vidljiva je plava strana figurice). Operator vraća bool, i to true ako je figuricu moguće dovesti u inicijalni položaj, a false inače. Dovođenje na inicijalni položaj mora koristiti ukupno najmanji mogući broj poziva gore navedenih operatora. (Uočite da unutar funkcije operator! smijete deklarirati druge varijable tipa Ploca i na njima pozivati operatore; ti se pozivi ne broje.) **Ovaj operator mora biti član klase Ploca.**

Napomene:

- Operatore (osim !, ++, --, >>) smijete definirati bilo kao članove, bilo kao friend funkcije klase Ploca.
- 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 figurice biti moguće dovesti u inicijalni položaj pomoću **samo 4 ili manje** poziva operatora ++, -- i >>. U oba testna primjera, operator ! će biti pozvan za 3 različite ploče.
- 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 "ploca.h"

using namespace std;

class TestPloca : public Ploca
{
public:
    int brojPoziva;
    TestPloca( int N ) : Ploca( N ) {}
    TestPloca &operator>>( int phi )
    {
        ++brojPoziva;
        Ploca::operator>>( phi );
        return *this;
    }
};


int main( void )
{
    TestPloca P( 4 );
    // PP..
    // ....
    // ....
    // ....

    cout << P( make_pair( 3, 4 ) ) << endl; // stavi kamencic, ispise 1
    // PP..
    // ....
    // ...x
    // ....

    pair<int, int> x = *P;
    cout << x.first << ", " << x.second << endl; // 1, 1
    cout << ~P << endl; // "plava"

    ++P;
    // ..CC
    // ....
    // ...x
    // ....

    P--;
    // ....
    // ..PP
    // ...x
```

```
    // ....

    P--; // ne moze, tamo je kamencic
    // ....
    // ..PP
    // ...x
    // ....

    P >> 90;
    // ....
    // ..P.
    // ..Px
    // ....

    --P;
    // ....
    // .C..
    // .C.x
    // ....

    P.brojPoziva = 0;
    !P; // napravi P > 90, pa P++
    // ovo treba ispisati 1 (u testnom mainu ce se brojati svi operatori, pa ce ispisati 2):
    cout << P.brojPoziva << endl;
    // PP..
    // ....
    // ...x
    // ....

    Ploca Q( 4 );
    cout << Q( make_pair( 3, 4 ) ) << endl; // ispise 1
    cout << Q( make_pair( 1, 1 ) ) << endl; // ispise 2
    cout << Q( make_pair( 1, 3 ) ) << endl; // ispise 1
    // PPx.
    // ....
    // ...x
    // ....

    cout << ( P == Q ) << endl; // 1

    return 0;
}
```