`Boardgame` è un gioco da tavolo che ha luogo su di una scacchiera 6x6 e vede fronteggiarsi due giocatori, il bianco ed il nero, con i loro rispettivi eserciti. Ogni casella della scacchiera è individuata dalla coppia (i, j), dove i è un carattere da A ad F indicante la riga e j è un numero da 1 a 6 indicante la colonna. In ogni casella, inoltre, può trovarsi un soldato bianco, un soldato nero o nessun soldato. Implementare le seguenti operazioni che possono essere effettuate in una `Boardgame`.

--- PRIMA PARTE ---

✓ `Boardgame b;`
Costruttore di default che inizializza un `Boardgame b`. In principio ogni è esercito è formato da sei soldati disposti come segue: tre soldati bianchi nelle colonne pari della riga A, tre soldati bianchi nelle colonne dispari della riga B, tre soldati neri nelle colonne pari della riga E, tre soldati neri nelle colonne dispari della riga F. Vedi la figura di esempio nell'operatore di uscita.

✓ `b.muovi(id, jd, is, js);`
Operazione che permette di muovere un soldato dalla casella `(is, js)` alla casella `(id, jd)`. Gli spostamenti consentiti all'interno della scacchiera sono in orizzontale o verticale di una o due caselle, oppure in diagonale di una sola casella. Sono inoltre proibiti (e pertanto la scacchiera rimane inalterata in questi casi) spostamenti che attraversino o terminino in caselle occupate da un altro soldato, spostamenti di lunghezza nulla, ovvero `(is, js)==(id, jd)`, e spostamenti nella cui casella di partenza non sia presente alcun soldato.

✓ `b.attacca(id, jd, is, js);`
Operazione che permette al soldato nella casella `(is, js)` di attaccare quello nella casella **adiacente** `(id, jd)`. Adiacenti sono le caselle di distanza uno in direzione orizzontale, verticale o diagonale. Sono da considerarsi proibite (nei quali casi la funzione deve lasciare inalterata la scacchiera) gli attacchi dove almeno una delle due caselle non contiene un soldato e quelle in cui un soldato ne attacca uno del suo stesso esercito. Qualora l'operazione vada a buon fine, il soldato attaccato viene rimosso dalla scacchiera e quello attaccante ne prende il posto.

✓ `cout << b;`
Operatore di uscita per il tipo `Boardgame`. I soldati bianchi vengono riportati con una "O" maiuscola mentre i neri con una "X" maiuscola. L'uscita per una scacchiera ad inizio partita ha la forma seguente:

```
  1 2 3 4 5 6
F|X   X   X  |
E|  X   X   X|
D|            |
C|            |
B|O   O   O  |
A|  O   O   O|
```

A separare ogni colonna c'è **un solo spazio vuoto**.

--- SECONDA PARTE ---

✓ `b<<n;`
Operatore di shift a sinistra che modifica la `BoardGame`, copiando la colonna di indice `i` nella colonna di indice `i-n` in modo circolare. Ad esempio, per `n` = 1 si ha che la colonna 5 viene copiata nella 4, la 4 nella 3 e così via. La 1 verrà copiata nella 6 mentre la 6 nella 5. L'operatore deve ammettere solo shift positivi.

✓ `b.compatta();`
Operazione che modifica la `BoardGame b` "compattandola" in basso a destra, cioè spostando tutte le righe vuote in alto e tutte le colonne vuote a sinistra.

✓ `~b;`
Operatore di complemento bit a bit che restituisce la differenza tra la *pressione* esercitata dal giocatore bianco e quella esercitata dal giocatore nero. La pressione esercitata da un giocatore è definita come il numero di soldati nemici sotto minaccia di attacco da parte di almeno due soldati.

Mediante il linguaggio C++, realizzare il tipo di dato astratto definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc.

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

int main(){

    cout<<"--- PRIMA PARTE ---"<<endl;
    cout<<"Test del costruttore"<<endl;
    Boardgame b;
    cout<<b<<endl;

    cout<<"Test della muovi"<<endl;
    b.muovi('D',3,'B',3);
    cout<<b<<endl;

    cout<<"Test della attacca"<<endl;
    b.attacca('D',3,'E',2);
    cout<<b<<endl;

    cout<<"---SECONDA PARTE---"<<endl;

    cout<<"Test operatore <<"<<endl;
    cout<<(b<<1)<<endl;

    cout<<"Test della compatta"<<endl;
    cout<<!b<<endl;

    cout<<"Test operatore ~"<<endl;
    b.muovi('B',3,'A',3);
    b.muovi('B',1,'A',1);
    cout<<b<<endl;
    cout<<(~b)<<endl;

}
```

USCITA ATTESA

```
--- PRIMA PARTE ---
Test del costruttore
  1 2 3 4 5 6
F|X   X   X  |
E|  X   X   X|
D|            |
C|            |
B|O   O   O  |
A|  O   O   O|

Test della muovi
  1 2 3 4 5 6
F|X   X   X  |
E|  X   X   X|
D|    O       |
C|            |
B|O       O  |
A|  O   O   O|

Test della attacca
  1 2 3 4 5 6
F|X   X   X  |
E|      X   X|
D|    X       |
C|            |
B|O       O  |
A|  O   O   O|
```

```
---SECONDA PARTE---
Test operatore <<
  1 2 3 4 5 6
F|  X   X   X|
E|    X   X  |
D|  X        |
C|            |
B|      O   O|
A|O   O   O  |

Test della compatta
  1 2 3 4 5 6
F|            |
E|  X   X   X|
D|    X   X  |
C|  X        |
B|      O   O|
A|O   O   O  |

  1 2 3 4 5 6
F|            |
E|  X   X   X|
D|    X   X  |
C|  X        |
B|O   O O   O|
A|        O  |

Test operatore ~
1
```