Es. 11.1 – Palazzo

Un `Palazzo` è composto da un numero massimo di piani e ogni piano contiene finestre. Il piano `j` contiene `j` finestre (i piani sono numerati a partire da 1). Ogni finestra può essere aperta o chiusa. Un Palazzo può avere anche un numero elevato di piani, quindi si richiede di risparmiare memoria nel caso di palazzi grandi. Implementare le seguenti operazioni che possono essere effettuate su un `Palazzo`:

--- PRIMA PARTE --- *(qualora siano presenti errori di compilazione, collegamento o esecuzione in questa parte, **l'intera prova sarà considerata insufficiente** e pertanto **non sarà corretta**)*

• `Palazzo p(N);`
Costruttore che inizializza un `Palazzo p` di al più `N` piani. Inizialmente `p` contiene un solo piano e la sua finestra è chiusa.

• `p(p1);`
Costruttore di copia che inizializza il `Palazzo p` uguale al `Palazzo p1`.

• `p.aggiungi();`
Operazione che aggiunge un piano a `p`, le cui finestre sono tutte chiuse. Se il palazzo contiene già il numero massimo di piani, la funzione lascia il palazzo inalterato.

• `p.stampa();`
Operazione di uscita che stampa il numero di piani del palazzo e, per ogni piano, stampa lo stato di tutte le finestre secondo il formato seguente:

```
<3>
Piano 1: Aperta
Piano 2: Chiusa Aperta
Piano 3: Chiusa Chiusa Chiusa
```

In questo esempio, il palazzo `p` ha 3 piani. Il primo piano ha una finestra chiusa, il secondo piano ha una finestra chiusa ed una aperta, il terzo piano ha tre finestre chiuse.

--- SECONDA PARTE ---

• `~Palazzo();`
Distruttore per il tipo `Palazzo`.

• `p.cambia(i,j);`
Funzione che cambia lo stato della finestra `j` del piano `i` del palazzo `p`. I piani e le finestre sono numerati a partire da 1.

• `!p;`
Operatore di negazione logica che restituisce il numero totale di finestre aperte del palazzo `p`.

• `p%=p1;`
Operatore di modulo e assegnamento che modifica il palazzo `p` come segue: se `p` e `p1` hanno numero di piani uguale, chiude le finestre aperte di `p` che sono chiuse in `p1`. Se `p` e `p1` hanno numero di piani diverso, l'operatore lascia `p` inalterato.

Mediante il linguaggio C++, realizzare il tipo di dato astratto `Palazzo`, definito dalle precedenti specifiche. Individuare le eventuali situazioni di errore e metterne in opera un corretto trattamento.

Il main di prova è il seguente:

```
#include "compito.h"
#include <iostream>
using namespace std;

int main()
{
  // PRIMA PARTE:
  cout << "Test del costruttore:" << endl;
  Palazzo p1(5);
  p1.stampa();
  cout << endl;

  cout << "Test del costruttore di copia:" << endl;
  Palazzo p2(p1);
  p2.stampa();
  cout << endl;

  cout << "Test della aggiungi:" << endl;
  p1.aggiungi();
  p1.aggiungi();
  p1.stampa();
  cout << endl;

  // SECONDA PARTE:
  /*cout << "Test del distruttore:" << endl;
  {
      Palazzo p(20);
  }
  cout << "(p e' stato distrutto)" << endl;

  cout << endl << "Test della cambia:" << endl;
  p1.cambia(2, 1);
  p1.cambia(3, 3);
  p1.stampa();
  cout << endl;

  cout << "Test operator! :" << endl;
  cout << !p1 << endl;

  cout << endl << "Test operator%= :" << endl;
  Palazzo p3(5);
  p3.aggiungi();
  p3.aggiungi();
  p1 %= p3;
  p1.stampa();
  cout << endl;*/

  return 0;
}
```

L'output desiderato del main di prova è il seguente:

```
Test del costruttore:
<1>
Piano 1: Chiusa

Test del costruttore di copia:
<1>
Piano 1: Chiusa
```

```
Test della aggiungi:
<3>
Piano 1: Chiusa
Piano 2: Chiusa Chiusa
Piano 3: Chiusa Chiusa Chiusa

Test del distruttore:
(p e' stato distrutto)

Test della cambia:
<3>
Piano 1: Chiusa
Piano 2: Aperta Chiusa
Piano 3: Chiusa Chiusa Aperta

Test operator! :
2

Test operator%= :
<3>
Piano 1: Chiusa
Piano 2: Chiusa Chiusa
Piano 3: Chiusa Chiusa Chiusa
```