`CittaIlluminate` è un innovativo sistema regionale di illuminazione cittadina. Esso può gestire un numero illimitato di città. Ogni città è caratterizzata da un nome, che è una stringa non vuota di al più 20 caratteri, e da un reticolato `NxN` di luci, con `N>0` che varia da città a città. Ogni luce può essere accesa o spenta.

Implementare le seguenti operazioni che possono essere effettuate su `CittaIlluminate`:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

✓ **`CittaIlluminate ci;`**

Costruttore di default che inizializza un `CittaIlluminate`, inizialmente vuoto.

✓ **`cout << ci;`**

Operatore di uscita per il tipo `CittaIlluminate`, che stampa a schermo la lista di città ed il loro stato di illuminazione, nel seguente formato:

```
- Livorno
  A S A
  S S S
  A S S
  Accese 3/9
- Firenze
  S S S S
  A S A S
  S S A A
  S A S S
  Accese 5/16
```

In particolare, per ogni città, vengono indicate quali luci del reticolato sono accese (`A`) e quali spente (`S`). Inoltre, viene indicato il numero di luci attualmente accese sul numero di luci totali. Le città devono essere stampate in ordine di inserimento. Per esempio, in `CittaIlluminate` di cui sopra, prima è stata inserita "Livorno" e poi "Firenze".

✓ **`ci.aggiungi(nome, dim);`**

Metodo che aggiunge al `CittaIlluminate ci` una nuova città caratterizzata da `nome` e da un reticolato di luci avente lato di lunghezza `dim`. La città ha inizialmente tutte le luci spente. In caso di input errati, la città non viene aggiunta e `ci` rimane invariato. La stessa cosa accade qualora una città con stesso nome (*case sensitive*) fosse già presente.

✓ **`ci.commuta(nome, riga, colonna);`**

Metodo che commuta la luce con coordinate `riga` e `colonna` nella città caratterizzata da `nome`, cioè la spegne se era accesa e la accende se era spenta. Righe e colonne nel reticolato sono numerate a partire da 1. In caso di input errati, `ci` rimane invariato.

✓ **`CittaIlluminate ci2 = ci;`**

Costruttore di copia che crea `CittaIlluminate ci2`, inizializzandola uguale a `ci`.

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

- ✓ **`~CittaIlluminate();`**

Qualora sia necessario, implementare il distruttore.

- ✓ **`ci.rimuovi(nome);`**

Metodo che rimuove da `ci` la città `nome`, solo qualora questa abbia tutte le luci spente. Se la città con quel nome non dovesse avere tutte le luci spente o se la città non fosse presente, `ci` rimane inalterato.

- ✓ **`~ci;`**

Operatore di complemento bit a bit, che restituisce un nuovo `CittaIlluminate` avente le stesse città con gli stessi nomi, ma nel quale lo stato delle luci di tutte le città è invertito, ovvero le luci spente vengono accese, quelle accese vengono spente.

- ✓ **`!ci;`**

Operatore di negazione logica per `CittaIlluminate`, che modifica `ci` spegnendo tutte le luci delle sole città che hanno almeno il 20% di luci accese. Ad esempio, se Pisa ha 2 luci accese su 9, tali luci verranno spente. Se invece Firenze ha 3 luci accese su 16, quelle luci rimarranno accese. L'operatore restituisce un riferimento all'oggetto stesso.

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

Uscita che deve produrre il programma

```
--- PRIMA PARTE ---
Test costruttore e funzione aggiungi
- Pisa
  S S S
  S S S
  S S S
  Accese 0/9
- Viareggio
  S S
  S S
  Accese 0/4
- Firenze
  S S S S
  S S S S
  S S S S
  S S S S
  Accese 0/16

Test funzione commuta
- Pisa
  S S S
  S A S
  S S A
  Accese 2/9
- Viareggio
  S S
  S S
  Accese 0/4
- Firenze
  A S S S
  S S S S
  S A A S
  S S S S
  Accese 3/16

Test costruttore di copia
- Pisa
  S S S
  S A S
  S S A
  Accese 2/9
- Viareggio
  S S
  S S
  Accese 0/4
- Firenze
  A S S S
  S S S S
  S A A S
  S S S S
  Accese 3/16
- Livorno
  S S S
  S S S
  S S S
  Accese 0/9
```

```
--- SECONDA PARTE ---
Test eventuale distruttore
Distruttore chiamato

Test funzione rimuovi
- Pisa
  S S S
  S A S
  S S A
  Accese 2/9
- Firenze
  A S S S
  S S S S
  S A A S
  S S S S
  Accese 3/16

Test operatore di negazione logica
- Pisa
  S S S
  S S S
  S S S
  Accese 0/9
- Firenze
  A S S S
  S S S S
  S A A S
  S S S S
  Accese 3/16

Test operatore di complemento
- Pisa
  A S A
  A A A
  A A A
  Accese 8/9
- Firenze
  S A A A
  A A A A
  A S S A
  A A A A
  Accese 13/16
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.
In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.