Il tipo astratto `Metropolitana` rappresenta una versione molto semplificata di una vera metropolitana. La semplificazione principale consiste nel fatto che una `Metropolitana` è formata solamente da cinque stazioni (identificate con numeri da 1 a 5) e un'unica linea. Ad ogni stazione possono essere presenti utenti in fila ma non possono ammontare a più di 50. Inoltre, presso ogni stazione può essere presente al più un treno. Ciascun treno è identificato da due interi: la sua capienza massima (positiva ma minore di 100) e il numero di posti liberi rimanenti a disposizione. Le varie stazioni possono essere collegate tra loro o meno mediante tunnel. Tali tunnel sono bidirezionali: se la stazione 1 è connessa alla 2, allora vale anche il viceversa. Un treno può muoversi da una stazione all'altra solo se queste sono collegate tra loro. Implementare le seguenti operazioni che possono essere effettuate su una `Metropolitana`:

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

- ✓ **`Metropolitana m;`**

Costruttore di default che inizializza una `Metropolitana`. Inizialmente le stazioni sono tutte sconnesse tra di loro, non vi sono treni e non vi sono utenti in attesa.

- ✓ **`m.aggiungi_utenti(quanti, stazione);`**

Funzione membro che aggiunge `quanti` utenti in attesa presso la stazione `stazione`. Qualora vi sia presente un treno con posti liberi, gli utenti provvederanno a salirvi immediatamente. La funzione restituisce il numero complessivo di utenti che sono effettivamente riusciti ad entrare nella `Metropolitana m`. In caso di input scorretti esso ammonta a 0 (nessun utente sale sul treno).

- ✓ **`m.aggiungi_connessione(s1, s2);`**

Funzione membro che aggiunge una connessione tra le due stazioni `s1` ed `s2`. L'implementazione deve permettere la concatenazione della funzione, ovvero permettere di eseguire il codice `m.aggiungi_connessione(s1,s2).aggiungi_connessione(s3,s4)`. Inoltre, non possono esistere connessioni di una stazione con sé stessa.

- ✓ **`m.aggiungi_treno(capienza, stazione);`**

Funzione membro che aggiunge un nuovo treno con capienza pari a `capienza` persone presso la stazione `stazione`. In caso l'operazione abbia esito positivo, il treno provvede immediatamente a caricare tutti gli utenti possibili che sono attualmente in fila presso la stazione corrente. Una volta eseguita questa operazione, la funzione restituisce `true`. In caso l'operazione non abbia successo, la funzione restituisce `false` e lascia inalterata la struttura dati.

- ✓ **`cout << m;`**

Operatore di uscita per il tipo `Metropolitana`. La stampa a video ha la seguente struttura:

```
 5  X  X
 0 15 10
16 20  0
 3  X  X
10 10  0

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5
```

La prima parte della rappresentazione consiste in una matrice 5x3 rappresentante lo stato delle cinque stazioni. Con riferimento all'esempio sopra, la semantica è la seguente: la prima stazione ha 5 utenti in attesa e non vi è nessun treno in sosta; la seconda stazione non ha alcun utente in attesa, vi è un treno in sosta con

una capienza di 15 posti di cui 10 sono liberi; la terza stazione vede 16 utenti in attesa, vi è un treno in sosta con una capienza di 20 persone ma non può caricarne altre perché è pieno (ha 0 posti liberi rimanenti); la descrizione delle stazioni quattro e cinque è omessa perché ridondante.

La seconda parte della rappresentazione consiste in un'unica riga descrivente le connessioni esistenti tra le stazioni. Nel caso in esempio, la stazione 1 è connessa alla 2 e alla 4, la stazione 3 è connessa alla 4, quest'ultima è connessa anche alla 5 (valgono i viceversa).

La stampa a video della prima parte della rappresentazione ha il seguente formato: ogni elemento della matrice occupa 2 caratteri, tra ogni colonna della matrice vi è 1 solo spazio. Qualora sia sufficiente utilizzare un solo carattere (ad esempio nella prima riga il valore 5 o le "X"), provvedere ad un allineamento a destra. Si noti che le "X" sono maiuscole.

La stampa a video della seconda parte della rappresentazione ha il seguente formato: la stazione con numero più piccolo in una connessione è sempre scritta a sinistra, il simbolo "<->" dista da ogni numero di stazione un solo spazio, tra due connessioni vi sono 2 spazi, le connessioni sono ordinate per stazioni crescenti come segue: `1 <-> 2` precede `1 <-> 4` che precede 2 `<-> 3` che precede 2 `<-> 4`.

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

✓ **`m.muovi_treno(quanti_scendono, s1, s2);`**

Funzione membro che muove il treno dalla stazione `s1` alla stazione `s2`. Una volta giuntovi, provvede a fare scendere `quanti_scendono` persone attualmente a bordo e a caricare quanti più utenti in fila possibili. Una volta scese dal treno, le persone non entrano in fila in quanto sono state servite dalla `Metropolitana m` e appena scese vi escono. Il valore dell'intero `quanti_scendono` deve essere gestito per sostituzione: se negativo deve essere considerato 0; se eccede il numero di passeggeri a bordo del treno deve essere considerato pari al numero di passeggeri correnti. In tutte le altre situazioni di errore la funzione lascia la struttura dati inalterata e restituisce `false`, altrimenti `true`.

✓ **`m.rimuovi_treno(stazione);`**

Funzione membro che rimuove il treno servente presso la stazione `stazione` della `Metropolitana m`. Per farlo, tutti gli attuali passeggeri devono essere fatti scendere presso la corrente stazione. Qualora ciò non sia possibile la funzione fallisce, lascia la struttura dati inalterata e restituisce `false`. In tutti i casi in cui la funzione non fallisce, deve modificare opportunamente la struttura dati e restituire `true`.

✓ **`!m;`**

Operatore di negazione logica che restituisce il bilancio della `Metropolitana m`. Con bilancio si intende la differenza tra il numero totale di utenti in attesa in tutte le stazioni e il numero di posti liberi tra tutti i treni.

✓ **`++m;`**

Operatore di incremento prefisso che modifica la metropolitana `m` aggiungendo, qualora vi sia posto, un utente in fila ad ogni stazione. Nel caso in cui in una stazione l'utente trovi un treno con posti liberi vi deve salire immediatamente.

Mediante il linguaggio C++, realizzare il tipo di dato astratto `Metropolitana`, 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 del costruttore
 0  X  X
 0  X  X
 0  X  X
 0  X  X
 0  X  X


Test della aggiungi_connessione
 0  X  X
 0  X  X
 0  X  X
 0  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

Test della aggiungi_utenti
15 0 0

22  X  X
13  X  X
 0  X  X
 0  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

Test della aggiungi_treno
1 0 0

 0 30  8
13  X  X
 0  X  X
 0  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

 0 30  8
13  X  X
 0 20 20
 0  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5
```

```
--- SECONDA PARTE ---

Test della muovi_treno
 0  X  X
 0 30  5
 0 20 20
 0  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

 0  X  X
 0  X  X
 0 20 20
 0 30  5
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

 0  X  X
 0  X  X
 0 20 20
 0 30  5
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

Test della rimuovi_treno
 0  X  X
 0  X  X
 0 20 20
25  X  X
 0  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5

Test dell'operatore !
5

Test dell'operatore ++
 1  X  X
 1  X  X
 0 20 19
26  X  X
 1  X  X

1 <-> 2  1 <-> 4  3 <-> 4  4 <-> 5
```

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 dai docenti**.