Esercizio 1 del 27 Marzo 2017 (Calcolatrice matriciale). Scrivere un programma che esegua una semplice operazione tra matrici di `float` e stampi il risultato. Le operazioni possibili sono tre:

- **Trasposizione** di una matrice `a[m][n]`. In questo caso il risultato è una matrice `b[n][m]` tale che `b[i][j]` = `a[j][i]` per ogni elemento `i, j`.
- **Somma** tra due matrici `a[m][n]` e `b[m][n]` di uguali dimensioni. In questo caso il risultato è una matrice `c[m][n]` tale che `c[i][j]` = `a[i][j]` + `b[i][j]` per ogni elemento `i, j`.
- **Prodotto tra due matrici** `a[m][n]` e `b[n][p]` (non necessariamente di uguali dimensioni). In questo caso il risultato è una matrice `c[m][p]` tale che $\texttt{c[i][j]} = \sum_{\texttt{r}=0}^{\texttt{n}-1} \texttt{a[i][r]} \cdot \texttt{b[r][j]}$ per ogni elemento `i, j`.

Il programma riceve come input la matrice `a`, seguita dall'operazione da fare, seguita da una seconda matrice `b` se l'operazione è una somma o un prodotto. Le matrici sono rappresentate nel modo usuale: due valori `int` che rappresentato il numero di righe e di colonne della matrice (massimo 10×10) seguiti dagli elementi della matrice disposti per riga. L'operazione è identificata da un `char`: `t` per la trasposizione, `+` per la somma e `*` per il prodotto. **Assumere che l'input del programma sia sempre corretto.**

Il programma deve stampare su `cout` il risultato dell'operazione (rappresentato come l'input), oppure `dimensioni non corrette` se le dimensioni delle matrici di input non permettono di effettuare l'operazione. Risolvere l'esercizio implementando le seguenti funzioni:

- `void leggiM(float *a, int &m, int &n)` che legge da `cin` una matrice di dimensione ignota, restituendo in `a` la matrice, in `m` il numero di righe ed in `n` il numero di colonne;
- `void stampaM(float *a, int m, int n)` che stampa su `cout` una matrice `a[m][n]`;
- `bool controllo_somma(int m1, int n1, int m2, int n2)` che ritorna `true` se e solo se una matrice di dimensione `m1`×`n1` si può sommare ad una matrice di dimensione `m2`×`n2`;
- `void somma(float *a, float *b, float *c, int m, int n)` che esegue la somma della matrice `a[m][n]` con `b[m][n]` e mette il risultato in `c`;
- `bool controllo_prodotto(int m1, int n1, int m2, int n2)` che ritorna `true` se e solo se una matrice di dimensione `m1`×`n1` si può moltiplicare con una matrice di dimensione `m2`×`n2`;
- `void prodotto(float *a, float *b, float *c, int m, int n, int p)` che esegue il prodotto della matrice `a[m][n]` con `b[n][p]` e mette il risultato in `c`;
- `void trasponi(float* a, float *b, int m, int n)` che traspone la matrice `a[m][n]` e mette il risultato in `b`.

Si possono utilizzare altre funzioni ausiliare per migliorare la chiarezza e la modularità del codice.

Correttezza: scrivere Pre e Post per tutte le funzioni elencate, dare un invariante per tutti i cicli della funzione `prodotto` e dimostrarne la correttezza.

Esempio: dato il seguente input

```
2 3
1.0 1.0 2.0
0.0 1.0 -3.0
*
3 3
1.0 1.0 1.0
2.0 5.0 1.0
0.0 -2.0 1.0
```

il programma calcola il prodotto di una matrice 2×3 con una 3×3, stampando come risultato la matrice 2×3

```
2 3
3.0 2.0 4.0
2.0 11.0 -2.0
```