Esercizio 3 del 29 Marzo 2017 (Pattern matching).

Scopo di questo esercizio è scrivere un programma che elenchi tutte le occorrenze di un *pattern* in un *testo*.

Il testo viene rappresentato come un array bidimensionale `char T[100][100]` dove ogni riga dell'array contiene una riga del testo. Le righe del testo sono di lunghezza variabile e terminano con il carattere speciale `'\0'` (`char` con codice ASCII 0, da non confondersi con la cifra `'0'`). I caratteri che compongono il testo possono essere lettere o spazi.

Il pattern è una sequenza di `char` di lunghezza variabile che termina con il carattere speciale `'\0'`, e può contenere lettere, spazi e caratteri "jolly" `'?'` (che corrispondono a "qualsiasi carattere"). Viene memorizzato nell'array `char P[100]`.

Diciamo che il pattern `P` occorre in posizione `i,j` se ogni carattere del pattern `P[k]` diverso dal jolly e dal carattere speciale `'\0'` è uguale a `T[i][j+k]`. Come d'uso in C++, contiamo le righe e le colonne di `T` partendo da `0`.

Viene dato un `main` che effettua la lettura di `P` e `T`, ed inserisce il numero di righe del testo nella variabile `int n_righe`. Il programma deve produrre tutte le occorrenze del pattern nel testo, ordinate per riga e poi per colonna. Ogni occorrenza `i,j` viene stampata nella forma `riga: i colonna: j`. Se non c'è nessuna occorrenza del pattern, il programma stampa `Pattern non trovato`.

Risolvere l'esercizio implementando la funzione `bool match(char* P, char* S)` che ritorna `true` se e solo se il pattern `P` occorre nell'array lineare `S` a partire dalla posizione `0`. Anche in questo caso `P` ed `S` terminano con il carattere speciale `'\0'`. Per esempio, se il pattern è `P = aba\0`:

- la chiamata `match(P, S)` con `S = ababbac\0` ritorna `true`;
- la chiamata `match(P, S)` con `S = bbbabac\0` ritorna `false` (l'occorrenza `aba` non è in posizione `0`);
- la chiamata `match(P, S)` con `S = ab\0` ritorna `false` (`S` è più corto del pattern).

Correttezza: scrivere un invariante per tutti i cicli della funzione `match` e dimostrarne la correttezza rispetto alla precondizione e alla postcondizione:

PRE: `P` è un pattern che termina con `'\0'`, `S` è una sequenza di `char` che termina con `'\0'`

POST: la funzione ritorna `true` se e solo se `P` occorre in `S` a partire dalla posizione 0

Esempio: dato il pattern `P = ?el` ed il testo `T` di 3 righe

```
nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura
che la diritta via era smarrita
```

il programma produce l'output

```
riga: 0 colonna: 0
riga: 0 colonna: 10
riga: 1 colonna: 20
```