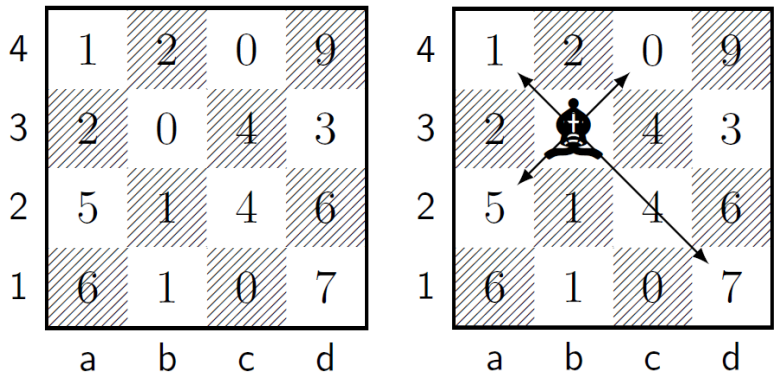
Esercitazione di laboratorio n. 3

(Le soluzioni verranno valutate in Laboratorio. Solo se caricate nella Sezione Elaborati del Portale entro e non oltre le 23:59 del 27/10/15 concorreranno all'assegnazione dei punti supplementari)

Esercizio n. 1: Alfiere
Tema d'esame del 02/09/2015 - Esercizio n°1 del percorso semplificato
Competenze: uso di matrici, problemi di selezione
Riferimento: Dal problema al programma: 4.5 Problemi di verifica e selezione

Si fornisce in input una possibile configurazione di una scacchiera quadrata NxN con riportati dei valori numerici per ogni cella.
Tale configurazione è caratterizzata da alcune celle a cui è associato valore nullo: tali posizioni sono attualmente libere, in attesa che un singolo alfiere venga posizionato.
Si scriva un programma che scelga in quale delle posizioni libere piazzare l'alfiere, così che la somma dei valori lungo le sue possibili direzioni di movimento sia massima.
Si ricorda che un alfiere muove solamente in diagonale rispetto alla posizione di partenza, sia in avanti sia in indietro.

Si consideri a titolo di esempio la seguente configurazione per una scacchiera 4x4:



La posizione migliore per l'alfiere è la cella b3, a cui si associa una somma complessiva pari a: 17.

Il file di input riporta sulla prima riga la dimensione N della scacchiera (N vale al massimo 10).
Le N righe successive contengono ognuna N interi a rappresentare i valori associati a ogni cella.
In relazione alla configurazione precedente, il file sarebbe:

```
4
1 2 0 9
2 0 4 3
5 1 4 6
6 1 0 7
```

Esercizio n. 2: Moltiplicazione in colonna
Tema d'esame del 16/06/2015 - Esercizio n°1 del percorso semplificato
Competenze: uso di matrici, problemi di codifica e problemi numerici
Riferimento: Dal problema al programma: 4.2.4 Problemi di codifica di numeri

Scrivere un programma che implementi una funzione del tipo

```
void mul (int *v1, int *v2, int n, int *v3);
```

che moltiplica due numeri interi di `n` cifre, il primo memorizzato nel vettore `v1`, il secondo nel vettore `v2`. La funzione restituisce il risultato della moltiplicazione nel vettore `v3`. In numeri sono memorizzati nei vettori in ragione di una cifra decimale per ciascun elemento. La funzione implementa l'algoritmo classico di moltiplicazione, operando per somma e scalamento e tenendo conto dei riporti, come illustrato dal seguente esempio:

```
      0 3 2 x
      2 4 3 =
------------
0 0 0 0 9 6 +
0 0 1 2 8   =
0 0 6 4
-----------
0 0 7 7 7 6
```

Si osservi che, per evitare overflow, il prodotto dovrà essere rappresentato su `2*n` cifre. Si suppone quindi che `v3` abbia dimensione `>= 2*n`.

Esercizio n. 3: Indicizzazione di riferimenti
Competenze: elaborazione di testi, ricerca in tabelle di nomi/stringhe
Riferimento: Dal problema al programma: 4.4 Problemi di text processing

Un file contiene un testo composto da un numero indefinito di righe, di lunghezza massima 80 caratteri ognuna.
All'interno del testo appaiono delle sequenze speciali, comprese tra due caratteri dollaro $, a rappresentare dei riferimenti che si possono ripetere più volte nel testo.
Si sappia che il numero massimo di riferimenti possibili è 20.
Si scriva una programma che analizzi il testo proposto, individuando tali sequenze, e assegnando ad ognuna di esse un indice progressivo. Nel far ciò, il programma deve riscrivere il testo su un secondo file sostituendo al riferimento incontrato l'indice ad esso assegnato racchiuso tra quadre, ossia nella forma [<indice>]. Si produca infine un terzo file contenente l'elenco di riferimenti affiancati dal rispettivo indice.
Nota: i riferimenti possono essere composti da più parole, ma non sono mai spezzati su più righe.

Esempio:
si abbia il seguente file:

```
$lorem$ $ipsum$ dolor $sit amet$, consectetur adipiscing elit. Sed viverra
porttitor diam id posuere. Pellentesque rhoncus vitae ex at consequat. Morbi
iaculis ut odio $sit amet$ venenatis. Donec quis lectus in diam pharetra
imperdiet et id $lorem$. In vel congue lectus. Maecenas aliquam quam quis
turpis consectetur accumsan. Ut id mauris lacinia, interdum mauris eu, mattis
```

```
odio. Nulla tempus in ligula $sit amet$ tincidunt. Donec dignissim neque
tempus, dapibus $tortor$ nec, sodales nisl. Suspendisse vel $tortor$ vitae
lacus
tristique aliquam in volutpat lectus. Nullam at ultricies diam, nec elementum
erat. Cras egestas, quam a tincidunt dapibus, $ipsum$ tellus placerat $ipsum$,
et gravida tellus ligula quis ante.
```

Il programma deve generare un nuovo file che rispetti il formato del precedente, come:

```
[1] [2] dolor [3], consectetur adipiscing elit. Sed viverra
porttitor diam id posuere. Pellentesque rhoncus vitae ex at consequat. Morbi
iaculis ut odio [3] venenatis. Donec quis lectus in diam pharetra
imperdiet et id [1]. In vel congue lectus. Maecenas aliquam quam quis
turpis consectetur accumsan. Ut id mauris lacinia, interdum mauris eu, mattis
odio. Nulla tempus in ligula [3] tincidunt. Donec dignissim neque
tempus, dapibus [4] nec, sodales nisl. Suspendisse vel [4] vitae lacus
tristique aliquam in volutpat lectus. Nullam at ultricies diam, nec elementum
erat. Cras egestas, quam a tincidunt dapibus, [2] tellus placerat [2],
et gravida tellus ligula quis ante.
```

Inoltre occorre generare l'indice dei riferimenti su un terzo file:

```
[1] lorem
[2] ipsum
[3] sit amet
[4] tortor
```

Esercizio n. 4: Analisi di puntatori
Competenze: tipo struct, matrici, definizione di puntatore
Riferimento: lezione di introdusione ai puntatori

Si consideri il seguente tipo struct:

```
typedef struct item_s {
  int a, b;
  char c;
  float d;
  char s[MAXS];
} Item;
```

Si scriva un programma che utilizzi una matrice di Item:

```
M[N][N];
```

N e MAXS siano definite mediante #define. Il contenuto della matrice viene acquisito da un file testo contenente NxN righe, ognuna delle quali contiene (separati da spazi) un Item: si consiglia una lettura formattata fscanf(fp, "%d%d %c%f%s", …)
Il programma acquisisca iterativamente da tastiera una coppia di interi r, c (compresi tra 0 e N-1). Visualizzi successivamente, per M[r][c], il contenuto e l'indirizzo di tutti i campi della relativa struct.