Esercitazione di laboratorio n. 3

(Caricamento sul portale entro le 23.59 del 31/10/2016 dell'esercizio 2 e di almeno uno tra gli esercizi 1 e 3)

Esercizio n. 1: Individuazione di regioni

Competenze: lettura/scrittura di file, manipolazioni di matrici statiche; puntatori e passaggio di parametri per riferimento (Puntatori e strutture dati dinamiche: 1.4)
Categoria: problemi di verifica e selezione (Dal problema al programma: 4.5)

Un file di testo contiene una matrice di interi (0 o 1) con il seguente formato:

- la prima riga del file specifica le dimensioni reali della matrice (numero di righe `nr` e numero di colonne `nc`). Si assuma che entrambi i valori siano al più pari a 50
- ciascuna delle `nr` righe successive contiene gli `nc` valori corrispondenti a una riga della matrice, separati da uno o più spazi
- ogni cella può contenere solamente o il valore 0 (associato al colore bianco) o il valore 1 (associato al colore nero)
- le celle nere sono organizzate in modo da formare regioni rettangolari discontinue (ogni regione nera è circondata da almeno una cornice di celle bianche, oppure dal bordo della matrice)
- l'adiacenza delle celle è considerata solo lungo i quattro punti cardinali principali (Nord, Sud, Ovest, Est), non in diagonale.

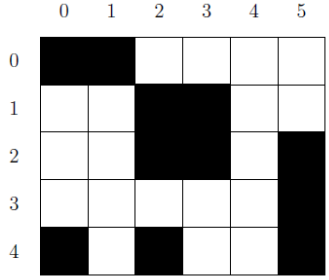
Si scriva un programma C che:

- legga tale matrice dal file di ingresso
- individui le regioni nere
- per ognuna regione produca in output le coordinate dell'estremo superiore sinistro e la lunghezza di base e altezza del rettangolo
- per ogni regione calcoli l'area totale espressa come numero di celle che la compongono.

Esempio:
contenuto del file:

```
5 6
1 1 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 1
0 0 0 0 0 1
1 0 1 0 0 1
```

configurazione corrispondente



Output del programma:

```
Regione 1: estr. sup. SX = <0,0> b = 2, h = 1, Area = 2
Regione 2: estr. sup. SX = <1,2> b = 2, h = 2, Area = 4
Regione 3: estr. sup. SX = <2,5> b = 1, h = 3, Area = 3
Regione 4: estr. sup. SX = <4,0> b = 1, h = 1, Area = 1
Regione 5: estr. sup. SX = <4,2> b = 1, h = 1, Area = 1
```

Requisiti:
Supponendo di avere dichiarato una matrice di interi `M` e di aver definito `MAXR` come `50` la matrice va acquisita mediante una funzione (`leggiMatrice`) che ne ritorna il numero di righe e di colonne effettivamente usati, come parametri "by reference" (o meglio, con puntatori by value). La funzione deve poter essere chiamata con un'istruzione del tipo:

```
leggiMatrice(M,MAXR,&nr,&nc);
```

Si chiede, per effettuare il riconoscimento delle regioni, di utilizzare una funzione `riconosciRegione` che, data una casella della matrice, determini se si tratti o meno di estremo superiore sinistro di una regione, ritornandone "by reference" (come per la precedente) le dimensioni del rettangolo, e avente come valore di ritorno un intero booleano (vero: rettangolo trovato, falso: rettangolo non trovato).
La funzione deve poter essere chiamata come segue:

```
if (riconosciRegione(M,nr,nc,r,c,&b,&h)) {
  // stampa messaggio per rettangolo con
  // estremo in (r,c), base b e altezza h
  ...
}
```

Esercizio n. 2: Log di accessi a server

Competenze: elaborazione di testi, ricerca in tabelle di nomi/stringhe, tipi enumerativi
Categoria: problemi di elaborazione testi mediante stringhe (Dal problema al programma: 4.4.3) e problemi di selezione (Dal problema al programma: 4.5.2)

Il web server di un'azienda mantiene un log degli accessi con il seguente formato:

<indirizzo IP> <user> <timestamp> <tipo richiesta> <risorsa> <risposta>

Tali informazioni sono memorizzate in un file testuale `accessi.txt` sulla cui prima riga è presente un singolo intero positivo che indica il numero di linee del file stesso (al più 1000 righe). Tutte le stringhe sono lunghe al massimo 30 caratteri.

Si scriva un programma C in grado di rispondere alle seguenti interrogazioni:

- elencare tutti gli utenti, per nome, che abbiano fatto accesso in un certo intervallo di date
- elencare tutti gli utenti, per IP, che abbiano fatto accesso in un certo intervallo di date
- elencare tutti gli utenti, per nome, per cui almeno una richiesta non sia stata accolta (risposta con valore 401)
- elencare tutte le risorse per cui sia stata effettuata almeno una richiesta in un certo intervallo di date.

Esempio:
contenuto del file di log:

```
8
178.1.192.33 user1 10/09/2016:13:12:55 GET index.html 200
```

```
34.52.1.37 user2 11/09/2016:12:09:42 GET private.html 401
178.1.192.33 user1 07/10/2016:16:34:36 GET intranet/login 401
178.1.192.38 user3 19/09/2016:00:57:01 GET extranet/login 200
126.189.12.12 user4 30/08/2016:11:11:10 GET services/list 200
130.192.165.91 user1 01/09/2016:11:01:12 GET services/list 200
130.192.165.92 user2 31/08/2016:03:29:07 GET admin/main.html 401
178.1.192.33 user1 14/08/2016:21:51:59 GET admin/main.html 200
```

Se l'intervallo di date è `31/08/2016 - 15/09/2016`:

- la risposta alla prima domanda è
  ```
  user1
  user2
  user1
  user2
  ```
- la risposta alla seconda domanda è:
  ```
  178.1.192.33
  34.52.1.37
  130.192.165.91
  130.192.165.92
  ```
- la risposta alla terza domanda è:
  ```
  user2
  user1
  user2
  ```
- la risposta alla quarta domanda è
  ```
  index.html
  private.html
  services/list
  admin/main.html
  ```

Requisiti:
Supponendo di aver definito `MAXN` come `1000 e` dichiarato un vettore di `struct Tabella`, di dimensione `MAXN`, in grado di contenere i dati presenti su file, la tabella va acquisita mediante una funzione (`leggiTabella`), che ne ritorna il numero di dati effettivamente letti come valore di ritorno. La funzione deve poter essere chiamata con un'istruzione del tipo:

```
nDati = leggiTabella(Tabella,MAXN);
```

Si chiede inoltre di definire un tipo enum `comando_e`, contenente i simboli `r_nome, r_IP, r_rifiuto, r_risorse, r_fine` (si veda il paragrafo 4.4.1, Dal problema al programma). Si realizzino poi per la ricerca e stampa:

- una funzione `leggiComando` che, letta una parola da tastiera riconosca le parole "nome", "IP", "rifiuto", "risorsa", "fine" e ritorni il valore corrispondente del tipo `comando_e`
- una funzione `selezionaDati` che, ricevuti come parametri la tabella e un comando, effettui la ricerca corrispondente alla richiesta, stampando i dati selezionati.

Dato `comando`, di tipo `comando_e` , la funzione `selezionaDati` deve poter essere richiamata come segue:

```
selezionaDati(Tabella,nDati,comando)
```

Esercizio n. 3: Cammino su una mappa

Competenze: lettura/scrittura di file, manipolazioni di matrici statiche; Puntatori e passaggio di parametri per riferimento (Puntatori e strutture dati dinamiche: 1.4), Matematica discreta
Categoria: problemi di verifica (Dal problema al programma: 4.5)

Un file di testo contiene una matrice di interi con il seguente formato:

- la prima riga del file specifica le dimensioni reali della matrice (numero di righe `nr` e numero di colonne `nc`). Si assuma che entrambi i valori siano comunque al più pari a 50
- ciascuna delle `nr` righe successive contiene gli `nc` valori (interi positivi, o nulli) corrispondenti a una riga della matrice, separati da uno o più spazi. Se il valore è 0, la cella non può far parte di un cammino. Se il valore è positivo, esso rappresenta il "peso" della cella in un cammino di cui eventualmente fa parte.

Si scriva un programma C che:

- legga tale matrice dal file di ingresso
- acquisisca da tastiera un cammino nella forma di coppie di interi <riga, colonna> su righe separate. Il cammino termina con la coppia <-1, -1>
- verifichi se si tratta effettivamente di un cammino, cioè se connette o meno cella di partenza (la prima) e cella di arrivo (l'ultima) passando per celle permesse
- in caso affermativo:
    - ne calcoli il peso, inteso come somma dei valori memorizzati nelle celle che lo compongono
    - verifichi se il cammino è semplice.

Sono ammessi movimenti verso tutte e otto (al più) celle adiacenti rispetto alla posizione corrente.

Esempio:
contenuto del file:

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

cammino da da verificare

```
0 0
0 1
0 2
1 2
2 1
3 2
-1 -1
```

risposta

```
il cammino è corretto, costa 12 ed è semplice
```

Requisiti:
Supponendo di avere dichiarato una matrice di interi `M` e di aver definito `MAXR` come `50` la matrice va acquisita mediante una funzione (`leggiMatrice`) che ne ritorna il numero di righe e

di colonne effettivamente usati, come parametri "by reference" (o meglio, con puntatori by value). La funzione deve poter essere chiamata con un'istruzione del tipo:

```
leggiMatrice(M,MAXR,&nr,&nc);
```

Si consiglia di effettuare la verifica del cammino, mentre se ne fa l'acquisizione da tastiera, usando una funzione `verificaCammino` che, ricevuta la matrice e acquisiti i dati sul cammino da tastiera, verifichi il cammino, ritornandone "by reference" (come per la precedente) il peso, e avente come valore di ritorno un intero booleano (vero: cammino corretto, falso: non è un cammino semplice).
La funzione deve poter essere chiamata come segue:

```
if (verificaCammino(M,nr,nc,&p)) {
  // stampa messaggio per cammino semplice di peso p
  ...
}
```