Esercitazione di laboratorio n. 2

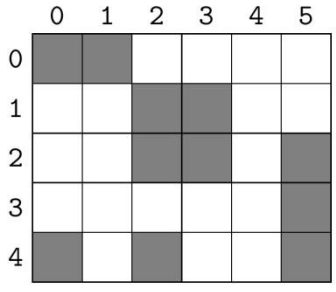
Esercizio n.1: **Individuazione di regioni**
Competenze: lettura/scrittura di file, manipolazioni di matrici;
Categoria: problemi di verifica e selezione (Dal problema al programma: 4.5)

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 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 (ogni regione nera è circondata da una cornice di celle bianche, oppure da bordo/i della matrice). A tal fine, si consideri che 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 la matrice dal file di ingresso (il file non contiene errori, quindi ci sono solo rettangoli neri che rispettano i vincoli)
- individui le regioni nere più grandi per altezza, larghezza e area totale
- in caso di parità, si riporti una tra le regioni individuate che soddisfano un certo criterio
- per ognuna di tali regioni produca in output le coordinate dell'estremo superiore sinistro e le sue caratteristiche (altezza, larghezza, area totale)

Esempio:	Mappa corrispondente:
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	

Output del programma:

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

Esercizio n.2: **Azienda di trasporti**
Competenze: selezione/filtro dati mediante ricerca in tabelle di nomi/stringhe, tipi enumerativi
Categoria: problemi di selezione (Dal problema al programma: 3.4.2 e 4.5.2)

Un'azienda di trasporti urbani traccia i propri automezzi in un file di log (file testuale di nome corse.txt).
Il file è organizzato come segue:

- sulla prima riga, un intero positivo che indica il numero di successive righe del file stesso (al più 1000)
- nelle righe successive le informazioni sulle tratte, uno per riga, con formato:

```
<codice_tratta><partenza><destinazione><data><ora_partenza><ora_
arrivo><ritardo>
```

Tutte le stringhe sono lunghe al massimo 30 caratteri. Il ritardo è un numero intero, eventualmente nullo, a rappresentare i minuti di ritardo accumulati dalla corsa.
Si scriva un programma C in grado di rispondere alle seguenti interrogazioni:

1. elencare tutte le corse partite in un certo intervallo di date
2. elencare tutti le corse partite da una certa fermata
3. elencare tutti le corse che fanno capolinea in una certa fermata
4. elencare tutte le corse che hanno raggiunto la destinazione in ritardo in un certo intervallo di date
5. elencare il ritardo complessivo accumulato dalle corse identificate da un certo codice di tratta

Le interrogazioni di cui sopra siano gestite mediante menu di comandi (si veda il paragrafo 4.4.1, Dal problema al programma). Ogni comando consiste di una parola tra "date", "partenza", "capolinea", "ritardo", "ritardo_tot" e "fine", eventualmente seguita sulla stessa riga da altre informazioni, ad esempio due date per "date", una fermata di partenza per "partenza", etc.
Si utilizzi la strategia di codifica dei comandi mediante tipo `enum comando_e`, contenente i simboli `r_date`, `r_partenza`, `r_capolinea`, `r_ritardo`, `r_ritardo_tot`, `r_fine`, che consente menu basati su `switch-case`.
Si consiglia di:

- realizzare una funzione `leggiComando` che, acquisito in modo opportuno il comando, ritorni il corrispondente valore di tipo `comando_e`
- realizzare una funzione `selezionaDati` che, ricevuti tra i parametri la tabella, la dimensione della tabella e il tipo di comando, gestisca mediante menu l'acquisizione delle informazioni aggiuntive necessarie per quel comando e la chiamata di un'opportuna funzione di selezione e stampa dei dati selezionati.

Esercizio n.3: **Occorrenze di parole**
Competenze: lettura/scrittura di file, manipolazioni di testi, ricerca in tabelle di nomi/stringhe
Categoria: problemi di elaborazione testi mediante stringhe (Dal problema al programma: 4.4.3)

Si scriva un programma in grado di localizzare, all'interno di un generico testo, le occorrenze di ogni parola contenente una certa sequenza di caratteri. Per parola si intende una sequenza di caratteri alfanumerici contigui (identificati dalla funzione `isalnum`), separata da spazi o segni di punteggiatura (identificati dalle funzioni `isspace` e `ispunct`).

Il programma riceve in input:

- il file `sequenze.txt`: sulla prima riga contiene il numero totale di sequenze, al più 20, sulle righe successive, una per riga, le sequenze da ricercare. La lunghezza delle singole sequenze è limitata a massimo 5 caratteri. Si trascuri la differenza tra maiuscole e minuscole
- il file `testo.txt`: contiene il testo. Il numero di righe non è noto a priori. Si assuma che la lunghezza di ogni riga sia al più pari a 200 caratteri. Si assuma inoltre che nessuna parola del testo sia più lunga di 25 caratteri.

Il programma deve visualizzare, per ognuna delle sequenze, quali parole la contengono e dove si trovano nel file. La posizione delle parole sia data in termine di conteggio delle parole dall'inizio del testo. Ai fini dell'esercizio ci si limiti a identificare e visualizzare solamente le prime 10 occorrenze per ogni sequenza.

Esempio

file `sequenze.txt`:

```
4
no
Al
per
s
```

file `testo.txt`:

```
Non sempre si capisce un esercizio alla prima lettura, ma prestando attenzione al testo e all'esempio non dovrebbe essere impossibile scrivere codice funzionante nonostante i dubbi iniziali. Se ancora non si capisce, allora basta chiedere all'esercitatore di turno.
```

La sequenza `no` è contenuta in `Non` (parola in posizione 1 nel testo), `non` (posizione 18), `nonostante` (posizione 25), `non` (posizione 31) e `turno` (posizione 40).

Esercizio n.4: **Valutazione di algoritmi di ordinamento**
Competenze: algoritmi di ordinamento iterativi, analisi empirica di complessità

Si considerino i seguenti algoritmi di ordinamento per ordinare in maniera ascendente vettori di interi:

- Selection Sort
- Insertion Sort
- Shell Sort

Si scriva un programma in C che per ogni sequenza numerica acquisita da file (`sort.txt`) invochi tutti gli algoritmi di ordinamento sopra indicati e stampi a video:

- il numero di scambi

- il numero di iterazioni del ciclo esterno
- per ogni passo del ciclo esterno il numero di iterazioni del ciclo interno
- il numero totale di iterazioni.

Il file `sort.txt` è caratterizzato dal seguente formato:

- sulla prima riga appare il numero `S` di sequenze numeriche
- seguono `S` righe nella forma `<lunghezza><sequenza>` dove `<lunghezza>` è un intero non negativo a rappresentare la lunghezza della sequenza riportata su tale riga, e `<sequenza>` sono `<lunghezza>` numeri separati da uno spazio.

Valutazione: l'esercizio 3 e uno tra 1 e 2 a scelta dello studente saranno oggetto di valutazione
Scadenza: caricamento di quanto valutato: entro le 23:59 del 27/10/2020.