Prova scritta del 8/1/2010

Strutturare adeguatamente i programmi ed evidenziarne la strutturazione mediante indentazione. Inserire anche adeguati commenti

1) Sia `Ind` il tipo di una struttura dati `struct` costituita da tre campi, `nome`, `cognome` e `email`, tutti e tre di tipo stringa (di lunghezza massima *80*).

(a) Realizzare una funzione booleana di nome `aggiungi` che, presi come suoi parametri un array `lista` di strutture di tipo `Ind`, il numero `n` di elementi in `lista`, ed una struttura `s` di tipo `Ind`, aggiunge `s` a `lista`, in posizione `n`, se non trova nessun elemento in `lista` con lo stesso valore dei campi `nome` e `cognome` di `s`; altrimenti aggiorna il campo `email` del (primo) elemento trovato con il valore del campo `email` di `s`. Nel caso `s` venga aggiunto a `lista` la funzione restituisce `true` come suo risultato e aggiorna il valore di `n` (`n` parametro per riferimento─obbligatorio); altrimenti restituisce `false`. Non è previsto che la funzione effettui controlli sul superamento della capacità dell'array `lista`.

(b) Descrivere la funzione anche tramite diagramma di flusso.

2) Sia `ml` un array di strutture di tipo `Ind` (dimensione massima *1000*) e `mailinglist.txt` e `nuovi.txt` due file contenenti terne di stringhe della forma

```
nome cognome indirizzo_di_posta_elettronica
```

Scrivere un programma principale che (1) legge tutti gli elementi del file `mailinglist.txt` e li memorizza nell'array `ml`; (2) legge tutti gli elementi del file `nuovi.txt` e per ciascuno di essi aggiorna `ml` utilizzando (obbligatoriamente) la funzione `aggiungi`; (3) scrive sullo std output il numero di elementi modificati ed il numero di elementi aggiunti a `ml`; (4) scrive su un nuovo file di nome `mailinglist_new.txt` tutti gli elementi (significativi) di `ml`, e quindi termina. N.B. Ogni terna nei file è separata dalla successiva da un "a capo" (l'ultima termina con *`end_of_file`*). Le stringhe nelle terne non contengono spazi o "a capo" e hanno una lunghezza massima di *80* caratteri. Si assume per semplicità che la dimensione di `ml` sia sufficiente a contenere tutti gli indirizzi letti dai due file di input.

3) Sia `M` una matrice di booleani di dimensioni *10 x 40* inizializzati a `false`. Scrivere un programma principale che: (1) richiede all'utente due coppie di indici (riga, colonna), $\langle x_1, y_1 \rangle$, $\langle x_2, y_2 \rangle$; (2) se almeno uno dei valori letti non è un indice valido per `M`, provvede a inviare su std output la matrice `M`, stampando un `'*'` per gli elementi di `M` con valore `true` e uno "spazio" per quelli con valore `false`, e quindi termina; (3) altrimenti, se $x_1 = x_2$ e $y_1 \leq y_2$, pone a `true` tutti gli elementi di `M` sulla riga x_1 compresi tra $M(x_1, y_1)$ e $M(x_2, y_2)$ (estremi inclusi), mentre se $y_1 = y_2$ e $x_1 \leq x_2$, pone a `true` tutti gli elementi di `M` sulla colonna y_1 compresi tra $M(x_1, y_1)$ e $M(x_2, y_2)$ (estremi inclusi); in entrambi i casi ripete da (1); (4) in tutti gli altri casi, semplicemente ripete da (1). N.B. La matrice viene stampata per righe, andando a capo al termine di ogni riga.