Prova scritta del 13/6/2012

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

1) (a) Realizzare una funzione di nome `maggiori` che preso un array di numeri reali `A`, la sua dimensione `n` ed un numero reale `s` calcola l'insieme `M` degli indici degli elementi di `A` che hanno valore > `s`. Es.: dato `A = {1.5,-3.0,1.7,1.2,0.7,1.4}` ed `s = 1.3` si avrà `M = {0,2,5}`. La funzione `maggiori` restituisce come suo risultato il numero di elementi memorizzati in `M`. N.B. Si definisca `M` come un array di interi da passare come ulteriore parametro alla funzione `maggiori`.
(b) Descrivere la funzione `maggiori` anche tramite diagramma di flusso.

2) Scrivere un programma che legge da un file, il cui nome è dato in input dall'utente (max *80* caratteri), una sequenza di numeri reali, rappresentanti le temperature misurate ad ogni ora per *7* giorni di una settimana, e li memorizza in una matrice *7 x 24* di nome `Temp`. Quindi, per ognuno dei *7* giorni della settimana, il programma determina, utilizzando obbligatoriamente la funzione `maggiori`, le ore del giorno in cui la temperatura risulta > di un valore `v` dato in input dall'utente (relativo all'intera settimana). I risultati calcolati vengono quindi stampati con il seguente formato (per ogni riga di stampa): *num. d'ordine del giorno – sequenza ore con temp. > v*).
N.B. Controllare l'esistenza del file. Controllare anche che i dati sul file siano completi (ovvero che non si incontri l'`end_of_file` prima di aver letto tutti i *7 x 24* dati). In entrambi i casi, in presenza di errore, terminare il programma con opportuno messaggio.

3) Sia `Ind` un tipo `struct` costituito da tre campi, `nome`, `cognome` e `email`, di tipo stringa (di lunghezza max *80*). Realizzare una funzione booleana di nome `add` che, presi come suoi parametri un array `v` di strutture di tipo `Ind`, il numero `n` di elementi in `v`, ed una struttura `s` di tipo `Ind`, aggiunge `s` a `v` in posizione `n` se non trova nessun elemento di `v` con stesso `nome` e `cognome` di `s`, altrimenti assegna al campo `email` dell'elemento di `v` con stesso `nome` e `cognome` di `s` il campo `email` di `s`. Nel caso `s` venga aggiunto a `v`, 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 `v`. Utilizzare soltanto stringhe *"tipo C"*.
Es., sia `v={{"aa","bb","ab@tin.it"}}`, `n=1`; se `s={"cc","bb","cb@gmail.com"}`, `v` diventa `{{"aa","bb","ab@tin.it"},{"cc","bb","cb@gmail.com"}}`, `n=2`; mentre se `s={"aa","bb","ab@gmail.com"}`, `v` diventa `{{"aa","bb","ab@gmail.com"}}`, `n=1`.
Realizzare anche un `main` di prova che legge da std input tre strutture di tipo `Ind`, le aggiunge a un array `A` di strutture di tipo `Ind` (di dimensione max. 20) utilizzando la funzione `add`, e quindi stampa `A`.