Prova scritta del 26/2/2009

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

1) Un multi-insieme è una collezione in cui ciascun elemento può apparire *n* volte ($n \geq 0$, *n* molteplicità dell'elemento). Un multi-insieme *M* può essere realizzato tramite due array, `A` e `B`: `A` contiene tutti gli elementi (distinti) di *M* e `B` contiene le molteplicità dei corrispondenti elementi di `A` (`B[i]` = molteplicità di `A[i]`). Ad esempio, il multi-insieme `[1,5,1,3,1,5]` è rappresentato dai due array `A=(1,5,3) e B=(3,2,1)`. N.B. Una molteplicità *0* in `B` significa che il corrispondente elemento in `A` non deve essere considerato (diciamo che è "libero").

(a) Scrivere una funzione di nome `somma` che, presi come suoi parametri due array di interi `A` e `B` che realizzano un multi-insieme di interi con la tecnica sopra descritta, ed il numero `k` di elementi in `A` e in `B`, calcola la somma di tutti gli elementi presenti nel multi-insieme (considerandoli con le loro molteplicità). Ad es., dato il multi-insieme rappresentato dai due array `A` e `B` di sopra, la funzione `somma` restituirà come risultato *16*.

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

2) Scrivere un programma che permetta di eseguire le seguenti operazioni su un multi-insieme *M* rappresentato con due array `Elem` e `Molt` (di dimensione max. *100*), secondo la tecnica descritta al punto *1*:

1. inserisci un elemento in *M* – richiede all'utente un intero `x`; se `x` non è presente in *M* inserisce `x` nel primo elemento "libero" di `Elem` (se non ci sono elementi "liberi", stampa opportuna segnalazione e continua); altrimenti, incrementa di *1* la molteplicità di `x`;
2. elimina un elemento da *M* – richiede all'utente un intero `x`; se `x` non è presente in *M*, stampa un'opportuna segnalazione e continua; altrimenti, decrementa di *1* la molteplicità di `x`;
3. somma – utilizzando la funzione `somma`, determina e stampa la somma degli elementi di *M*.
4. smetti.

Il programma presenta all'utente (su std output) il menù delle possibili operazioni, esegue l'operazione scelta e quindi ripete dall'inizio finché non viene scelta l'operazione 4. N.B. Si assuma che *M* sia inizialmente vuoto (e cioè l'array `Molt` deve essere inizializzato tutto a *0*). Definire nuove funzioni per operare su *M* ove opportuno (ad es., ricerca di un elemento con valore x, ricerca del primo elemento "libero").

3) Scrivere un programma che: richiede all'utente di fornire un intero `k` ed il nome di un file; legge dal file specificato una sequenza di stringhe, separate da *a capo*, fino a trovare la stringa `"stop"` o l'*end_of_file*; per ogni stringa letta determina la sua lunghezza `l`: se è `l` > `k` scrive la stringa su un file il cui nome è ottenuto da quello di input facendolo precedere dal prefisso `"stringhe_lunghe_"`, mentre se è `l` ≤ `k` la stringa viene scritta su un file con prefisso `"stringhe_corte_"` (ad es., se il nome del file di input è `"dati.txt"`, i nomi dei file di output saranno `"stringhe_lunghe_dati.txt"` e `"stringhe_corte_dati.txt"`). Si supponga che tutte le stringhe utilizzate siano stringhe qualsiasi (contenti anche spazi), di lunghezza massima *100*. N.B. Utilizzare soltanto stringhe tipo C (= array di caratteri …).

è memorizza le stringhe di l
per ogni stringa letta determina la sua lunghezza l e a seconda che sia l > k o l <= k scrive la stringa su un file il cui nome è ottenuto da quello di input facendolo precedereè memorizza le stringhe di l

3) Scrivere un programma che: legge da un file, il cui nome è dato in input dall'utente, una sequenza di stringhe, separate da a capo, fino a trovare la stringa "stop" o l'"end_of_file". Per ogni stringa letta determina la sua lunghezza: se è > è memorizza le stringhe di l

stringhe del C

Un multi-insieme è una collezione in cui ciascun elemento può apparire n volte (n >= 0, detta molteplicità dell'elemento). Un multi-insieme *M* può essere realizzato utilizzando due array A e B: A contiene tutti gli elementi (distinti) di *M* e un array B in cui ciascun elemento specifica la molteplicità del corrispondente elemento di A. Ad esempio, il multi-insieme [1,5,1,3,1,5] è rappresentato dai due array `A=(1,5,3) e B=(3,2,1)`. N.B. Una molteplicità 0 in B significa che il corrispondente elemento in A non deve essere considerato (diciamo che è "libero").

2) Si consideri un file di testo, di nome `"temperature.txt"`, contenente su ogni riga la temperatura minima, la temperatura massima ed una stringa rappresentante la data del giorno cui si riferiscono le due temperature (formato della data: `g/m/a`, con `g`, `m`, `a` numeri senza `0` iniziali; ad es., `10/2/2009`). Scrivere un programma principale che: (1) legge tutti i dati presenti nel file `"temperature.txt"` e li memorizza in un array `T` di strutture di tipo `S` (ogni riga del file in un diverso elemento di `T`*)*; (2) utilizzando (obbligatoriamente) la funzione `max_diff`, determina il giorno (ovvero l'elemento di `T`) in cui si è avuta la massima escursione termina (= differenza tra temperatura massima e minima); (3) stampa un messaggio con il seguente formato: "`Massima escursione termica: giorno` *`dd`* `del mese` *`mm`*", dove *`dd`* ed *`mm`* sono rispettivamente il numero del giorno ed il numero del mese ricavati dalla data presente nell'elemento di `T` individuato al punto (2).
N.B. Si utilizzino soltanto stringhe tipo C (= array di caratteri terminati da `'\0'`). i assuma che `T` abbia una dimensione massima di 1000.

Si assuma che gli array Elem e Mul abbiano dimensione massima 100 e che in caso
5.
3) Scrivere un programma principale che esegue ripetutamente alcune semplici operazioni su un multi-insieme M rappresentato con due array Elem e Mul secondo la tecnica descritta al punto 1. Le operazioni previste sono:
inserimento di un elemento x in M -- richiede all'utente un intero x; se x non è presente in M inserisce x nel primo elemento "libero" di Elem; altrimenti, incrementa di 1 la molteplicità di x;
eliminazione di un elemento x in M -- richiede all'utente un intero x; se x non è presente in M, stampa un'opportuna segnalazione; altrimenti, decrementa di 1 la molteplicità di x;
somma – utilizzando (obbligatoriamente) la funzione somma determina e stampa la sommadi tutti gli elementi di M.

Array pieno

crea una matrice di caratteri `M` di dimensioni *10 x 15* e la gestisce nel modo seguente: (1) inizializza tutti gli elementi di `M` con il carattere *"spazio*"; (2) richiede all'utente due interi `i` e `j` e scrive il carattere `'*'` in `M(i,j)` (se `i` o `j` sono maggiori delle dimensioni massime della matrice o uno dei due è negativo, dà opportuno messaggio e ripete l'input); (3) ripete il punto (2) finché `i` e j sono entrambi non negativi; (4) altrimenti, stampa la matrice per righe su standard output; (4) chiede all'utente se vuole continuare ed in caso affermativo ripete dal punto (1), altrimenti termina.