INFORMATICA 2010/11 - CdL in FISICA

PRIMO APPELLO – 21/06/2011

Scrivere **in stampatello** COGNOME, NOME e MATRICOLA su ogni foglio consegnato

N.B.: In tutti gli esercizi viene valutata anche la leggibilità del codice proposto. Non è consentito l'uso di istruzioni che alterino il normale flusso dell'esecuzione (come `continue`, `break` (tranne che in uno `switch`) e istruzioni di `return` all'interno di cicli che ne provochino l'uscita forzata).
Invece di usare interi come variabili booleane, utilizzare il tipo `boolean` definito da

`typedef enum {false, true} boolean;`

Le procedure/funzioni in generale non devono contenere alcuna istruzione di input/output (ad es. `scanf, printf, getchar, putchar,...`)

ESERCIZIO 1
Scrivere un programma C che legge una sequenza di numeri interi fornita dall'utente e ne stampa il massimo, il minimo e la media aritmetica (che in generale può essere un numero decimale). La sequenza termina quando l'utente immette uno zero, che è considerato far parte della sequenza. Il programma non deve utilizzare né un array né una struttura dati dinamica per memorizzare la sequenza. Il programma *può* (ma non deve) essere strutturato in modo da utilizzare una procedura o funzione ricorsiva; in questo caso la procedura o funzione può utilizzare istruzioni di input/output per interagire con l'utente.

ESERCIZIO 2
Scrivere una funzione C che ha come parametri formali un array di caratteri e la sua dimensione, e restituisce la posizione del primo carattere che compare almeno tre volte nell'array, ignorando la differenza tra maiuscole e minuscole. Se nessun carattere compare almeno tre volte, la funzione restituisce -1.

ESERCIZIO 3
Sia data la seguente procedura C:

```
void foobar(int x, int *y){
    x = *y;
    *y = *(y+1);
    *(y+1) = x;
}
```

Sia dato inoltre il seguente `main`:

```
int main(){
    int arr[] = {11, 22, 33, 44};
    foobar(arr[0], &arr[1]);
    foobar(arr[3], arr + 2);
}
```

1. Spiegare cosa fa la procedura `foobar`.
2. Indicare il contenuto dell'array `arr` dopo ognuna delle tre istruzioni del corpo del `main`, giustificando le risposte.
3. Cosa succede se si aggiunge come ultima istruzione del `main` la seguente?

```
foobar(arr[3], &arr[3]);
```

ESERCIZIO 4

Dato un insieme finito X, un *multinsieme su* X è una funzione $m\colon X \to \mathbb{N}$ che associa a ogni elemento di X la *molteplicità* con cui compare nel multinsieme.

Si vuole rappresentare un multinisieme m di caratteri come una lista contenente un nodo per ogni carattere che ha molteplicità non nulla in m. Ogni nodo contiene un carattere e la sua molteplicità, oltre al puntatore al nodo successivo.

N.B.: Si considerino diversi i caratteri minuscoli e maiuscoli.

(i) Si definiscano i tipi di dati necessari per implementare in `C` la rappresentazione indicata. Si chiami `MSet` il tipo di dati principale.

Si definiscano quindi le seguenti funzioni o procedure C che operano su oggetti di tipo `MSet`, rispettando il prototipo proposto:

(ii) `int mult (MSet m, char c);`
Restituisce la molteplicità del carattere `c` nel multinsieme `m`. Questa funzione *deve* essere definita in modo ricorsivo.

(iii) `void insert (MSet *p, char c, int k);`
Inserisce il carattere `c` con molteplicità `k` nel multinsieme puntato da `p`. Quindi se il carattere era già presente in un nodo, ne aumenta la molteplicità di `k`, altrimenti inserisce un nuovo nodo nella lista con il carattere `c` e con molteplicità `k`. La procedura non modifica il multinsieme se $k \leq 0$.

(iv) `void minus (MSet *m1, MSet m2);`
Modifica il multinsieme `m1` sottraendo `m2`, senza modificare `m2`. Dopo l'esecuzione della procedura, `m1` sarà il multinsieme definito come

$$(m1 - m2)(x) = \begin{cases} m1(x) - m2(x) & \text{se } m2(x) < m1(x) \\ 0 & \text{altrimenti} \end{cases}$$

(v) `MSet add (MSet m1, MSet m2);`
Senza modificare i multinsiemi `m1` e `m2`, questa funzione restituisce il multinsieme ottenuto come *somma* dei due argomenti, cioè il multinsieme in cui ogni carattere `c` ha come molteplicità la somma delle molteplicità in `m1` e `m2`.

N.B.: Nello svolgimento di questo esercizio **non** si può fare riferimento a funzioni/procedure viste a lezione o a esercitazione.