

	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione <b>INFORMATICA B</b> Appello 16 febbraio 2017		COGNOME E NOME				
	RIGA	COLONNA	MATRICOLA				
			Spazio riservato ai docenti <table border="1" style="float: right;"> <tr> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> </tr> </table>				

- Il presente plico contiene 3 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola.**
- Il tempo a disposizione è di 1 ora e 30 minuti.
- Non separate questi fogli. Scrivete la soluzione solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità. Cancellate le parti di brutta con un tratto di penna.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- È possibile scrivere a matita (e non occorre ricalcare al momento della consegna!).
- **È vietato utilizzare calcolatrici, telefoni, pc o altri apparecchi elettronici.** Chi tenti di farlo vedrà annullata la sua prova.
- È ammessa la consultazione di libri e appunti.
- **Qualsiasi tentativo di comunicare con altri studenti comporta l'espulsione dall'aula.**
- È possibile ritirarsi senza penalità.
- **Non è possibile lasciare l'aula conservando il tema della prova in corso.**
- L'esame **orale è parte integrante dell'esame** e deve essere realizzato **almeno sufficientemente** per il superamento dell'esame complessivo.
- Negli esercizi relativi a Matlab, verranno valutate positivamente solo le soluzioni che sfruttano le particolarità di Matlab per operare su matrici e vettori.

## Esercizio 1 (10 punti)

Sia  $\{x_t\}$ ,  $t = \{1, 2, \dots, T\}$  una serie storica, cioè una sequenza di valori che descrive, ad esempio, l'andamento del prezzo giornaliero di un titolo in borsa. Definiamo la **media mobile di  $\{x_t\}$  con finestra  $n$**  la serie storica  $\{m_t\}$ ,  $t = \{1, 2, \dots, T\}$  il cui valore nel punto  $t$  corrisponde alla media di  $x_t$  e degli  $n-1$  valori che precedono  $x_t$ . Nel caso in cui, prima di  $t$ , non vi siano almeno  $n$  valori, la media mobile corrisponde alla media di  $x_t$  e di tutti i valori che precedono  $x_t$ .

Ad esempio:

la media mobile con finestra 3 della serie storica

```
[10 2 3 13 101]
```

è:

```
[10 6 5 6 39]
```

in quanto

- 10 non ha valori che lo precedono nella serie
- 6 è la media di 2, 10
- 5 è la media di 3, 2, 10
- 6 è la media di 13, 3, 2
- 39 è la media di 101, 13, 3

- A. Implementare una funzione MATLAB `mediaMobile`, includendo il relativo help, che riceve in ingresso un vettore  $x$  contenente una serie storica e un intero  $n$  che rappresenta la dimensione della finestra da considerare, e restituisce un vettore contenente la media mobile di  $x$  con finestra  $n$ , con  $t = \{1, 2, \dots, \text{length}(x)\}$ .
- B. Implementare uno script MATLAB che:
- genera una serie storica, come un vettore  $x$  di 500 numeri casuali compresi fra 0 e 10;
  - legge da tastiera due valori interi  $a$  e  $b$ ;
  - memorizza nel vettore `mma` la media mobile di  $x$  con finestra  $a$  e nel vettore `mmb` la media mobile di  $x$  con finestra  $b$ ;
  - visualizza su un unico grafico la serie rappresentata dal vettore  $x$ , la media mobile `mma`, e la media mobile `mmb` (riportando sull'ascissa i valori fra 1 e 500);
  - memorizza in un vettore  $t$  tutti gli indici di  $x$  in cui le due medie mobili `mma` e `mmb` **si incrociano** (ovvero tutte le posizioni  $i$  in cui si ha `mma(i) >= mmb(i)` e `mma(i-1) < mmb(i-1)` o viceversa).

## Soluzione

A.

```
function mm = mediaMobile(x, n)
```

```
% mediaMobile: riceve il vettore x contenente una serie storica e un intero n che rappresenta la dimensione  
% della finestra da considerare per il calcolo della media mobile, e restituisce un vettore contenente la  
% media mobile di x in finestra n.
```

```
for i = 1:length(x)  
    j = max(1, i-n+1);  
    mm(i) = mean(x(j:i));  
end
```

B.

```
x = rand(1, 500) * 10;  
a = input('a = ');  
b = input('b = ');
```

```
mma = mediaMobile(x, a);  
mmb = mediaMobile(x, b);
```

```
plot(1:500, x, 1:500, mma, 1:500, mmb);
```

```
t = [find( (mma(2:end) >= mmb(2:end) & mma(1:end-1) < mmb(1:end-1)) |  
          (mma(2:end) < mmb(2:end) & mma(1:end-1) >= mmb(1:end-1)) )]
```

## Esercizio 2 (10 punti)

Si considerino i seguenti tipi in linguaggio C che descrivono i principali elementi di un famoso videogioco ambientato nel paese dei PocketMonster:

```
typedef enum{acqua, fuoco, erba, elettricita} Elementi;

typedef struct {
    int valoreAttributo;
    char nomeAttributo[50];
} Attributo;

typedef struct{
    Elementi tipo;
    int livello;
    Attributo attributi[4];
} PocketMonster;

typedef struct{
    PocketMonster gruppo[150];
    int nGruppo;
} PocketDex;
```

Un *PocketDex* include un vettore di *PocketMonster* contenente un numero di individui pari a *nGruppo* (fino a un massimo di 150) e ogni *PocketMonster* è caratterizzato dal suo elemento di appartenenza (acqua, fuoco, erba o elettricita), dal suo livello (un valore intero positivo) e da quattro attributi diversi. Ciascun attributo è caratterizzato da un nome e un valore, ad esempio *nomeAttributo* = "Attacco", *valoreAttributo* = 12. Gli attributi possono essere "Attacco", "Difesa", "Spirito critico", "Umorismo" e "Fedeltà".

Si supponga che sia stata dichiarata una variabile *dex* di tipo *PocketDex* e sia stata adeguatamente popolata.

- 1) Scrivere una porzione di codice in linguaggio C che cerchi tutti i *PocketMonster* di elemento acqua e aventi attributo "Attacco" con valore superiore a 20, e li copi (senza lasciare buchi) in un vettore *attaccantiForti* di tipo *PocketMonster*.
- 2) Scrivere un frammento di codice in linguaggio C che trovi il livello massimo dei *PocketMonster* presenti in *dex* in ciascuno degli elementi definiti nel gioco (acqua, fuoco, erba ed elettricita). Nel caso non vi sia in *dex* alcun *PocketMonster* per un dato elemento, il livello massimo per quell'elemento venga convenzionalmente posto pari al valore -1. Inoltre, il frammento di codice stampi a schermo il nome e livello massimo di ogni elemento, ad esempio:

```
Livello massimo dei PocketMonster presenti in dex:
Acqua: 34
Fuoco: 12
Erba: -1
Elettricità: -1
```

- 3) Definire in linguaggio C un nuovo tipo *Allenatore* che abbia un *nome* e un *cognome* (entrambi campi alfanumerici di 30 caratteri), una *squadra* di **al massimo 6** *PocketMonster* e **fino a 2** elementi preferiti tra acqua, fuoco, erba ed elettricita.

**N.B.** Non occorre scrivere un programma completo, ma semplicemente le porzioni di codice che svolgono le operazioni richieste, ricordandosi di dichiarare e inizializzare tutte le eventuali variabili ausiliarie necessarie.

## Soluzione

1)

```
PocketMonster attaccantiForti[150];
int i, j, nForti = 0;

for (i = 0; i < dex.nGruppo; i++){
    if(dex.gruppo[i].tipo == acqua){
        for(j = 0; j < 4; j++){
            if (strcmp(dex.gruppo[i].attributi[j].nomeAttributo,"Attacco") == 0
                && dex.gruppo[i].attributi[j].valoreAttributo > 20){
                attaccantiForti[nForti] = dex.gruppo[i];
                nForti++;
            }
        }
    }
}
```

2)

```
int maxLevel[4];
int i;

for (i = 0; i < 4; i++)
    maxLevel[i] = -1;

for (i = 0; i < dex.nGruppo; i++){
    if (dex.gruppo[i].livello > maxLevel[dex.gruppo[i].tipo])
        maxLevel[dex.gruppo[i].tipo] = dex.gruppo[i].livello;
}

printf("Livello massimo dei PocketMonster presenti in dex:\n");
printf("Acqua: %d\n" maxLevel[0]);
printf("Fuoco: %d\n" maxLevel[1]);
printf("Erba: %d\n" maxLevel[2]);
printf("Elettricit : %d\n" maxLevel[3]);
```

3)

```
typedef struct{
    char nome[30];
    char cognome[30];
    PocketMonster squadra[6];
    Elementi preferiti[2];
    int sizeSquadra;
    int sizePreferiti;
} Allenatore;
```

### Esercizio 3 (6 punti)

Si consideri il seguente programma in linguaggio C per calcolare il tempo medio di accesso alle informazioni in memoria in un sistema informatico dotato di memoria cache. Il programma legge da tastiera N diverse configurazioni di cache (HIT Time, HIT Rate e MISS Penalty) e le memorizza rispettivamente nei vettori HT, HR e MP, in modo tale che HT[i], HR[i] e MP[i] contengano le informazioni di HIT Time, HIT Rate e MISS Penalty relative alla stessa configurazione i-esima.

```
#include <stdio.h>
#define N 100

int main()
{
    int HT[N], MP[N], i;
    float HR[N];

    for (i = 0; i < N; i++)
        printf("\nInserisci Hit time: ");
        scanf("%d", &HT[i]);
        printf("\nInserisci Hit rate: ");
        scanf("%f", &HR[i]);
        printf("\nInserisci Miss penalty: ");
        scanf("%d", &MP[i]);

    // Inserire qui il frammento di codice richiesto

    return 0;
}
```

Rispondere a UNA delle seguenti due domande, a scelta:

- 1) Completare in linguaggio C il programma inserendo (dove indicato) un frammento di codice che individui la configurazione di cache (cioè la terna HT, HR e MP) a cui corrisponde il minor tempo medio di accesso. Stampare a video tale configurazione e il corrispondente tempo medio di accesso. Dichiarare e inizializzare anche eventuali variabili ausiliarie se necessario.
- 2) Scrivere in MATLAB uno script che svolge le stesse operazioni presenti nel programma in linguaggio C mostrato sopra (a meno del return 0) e completarlo inserendo, sempre in MATLAB, un frammento di codice che individui la configurazione di cache (cioè la terna HT, HR e MP), a cui corrisponde il minor tempo medio di accesso. Stampare a video tale configurazione e il corrispondente tempo medio di accesso.

### Soluzione

```
1)
float tmin;
int imin = 0;

tmin = HT[0] * HR[0] + MP[0] * (1 - HR[0]);
for (i = 1; i < N; i++)
    if (HT[i] * HR[i] + MP[i] * (1 - HR[i]) < tmin)
    {
        tmin = HT[i] * HR[i] + MP[i] * (1 - HR[i]);
        imin = i;
    }
```

```
printf("\nHT: %d; HR: %f; MP: %d => tempo medio di accesso: %f\n",HT[imin], HR[imin], MP[imin], tmin);
```

```
2)
N = 100;
for ii = 1:N
    HT(ii) = input('Inserisci Hit Time: ');
    HR(ii) = input('Inserisci Hit Rate: ');
    MP(ii) = input('Inserisci Miss Penalty: ');
end

t = HT .* HR + MP .* (1 - HR);
[tmin, index] = min(t);
fprintf('HT: %d; HR: %f; MP: %d => tempo medio di accesso: %f\n', HT(index), HR(index), MP(index), tmin);
```

#### Esercizio 4

Si supponga di avere a disposizione una funzione MATLAB chiamata merge che, dati due vettori contenenti valori ordinati in modo crescente, ne restituisce un terzo che contiene, sempre in ordine crescente, tutti i valori dei primi due vettori. Per esempio, il seguente frammento di codice:

```
v1 = [4 6 7 10 11 13];
v2 = [3 5 6 8 8 12];
r = merge(v1, v2);
```

risulterà in r pari a: [3 4 5 6 6 7 8 8 10 11 12 13]

Si consideri ora la seguente funzione ricorsiva che utilizza merge:

```
function [s] = MS(v)
    if (length(v) <= 1)
        s = v;
    else
        v1 = v(1:ceil(length(v)/2)); %ceil arrotonda un numero all'intero maggiore più vicino.
        v2 = v(ceil(length(v)/2)+1:end);
        vOut1 = MS(v1);
        vOut2 = MS(v2);
        s = merge(vOut1 vOut2);
    end
end
```

Si risponda alle seguenti domande:

1. Data la seguente chiamata della funzione: ris = MS([9 4 6 1 3]), qual è il valore che viene assegnato a ris?
2. Per motivare la risposta, si rappresentino e si commentino gli ambienti di esecuzione di tutte le chiamate ricorsive di MS che vengono eseguite nel caso descritto al punto 1.
3. In generale, qual è il compito svolto da MS?





