	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **ORMATICA B** 16 febbraio 2018		COGNOME E NOME
	Riga	Colonna	MATRICOLA
TEMA A			Spazio riservato ai docenti

- Il presente plico contiene 3 esercizi e 2 domande e **deve essere debitamente compilato con cognome e nome, e numero di matricola**.
- Il tempo a disposizione è di 2 ore.
- 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 (o ripudiate) 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 o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- **È vietata la consultazione di libri e appunti e qualsiasi altra risorsa**.
- **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**.
- **Per il superamento dell'esame è necessario dimostrare sufficienti competenze sia in C sia in Matlab, e quindi saper impostare correttamente esercizi in entrambi i linguaggi**.
- **MOLTO IMPORTANTE**: **risposte poco leggibili** (scritte molto piccolo, con calligrafia poco comprensibile, o molto disordinate) **non saranno considerate nella valutazione**.

Esercizio 1 (10 punti)

In una gara di nuoto, **N** giudici (al più 10) esprimono ognuno il proprio giudizio (**1**: buono, o **0**: scarso) su **K** nuotatori (al più 100). Il punteggio complessivo di ogni nuotatore (ognuno identificato da un numero progressivo) è un valore numerico tra 0 e 1 calcolato come media dei giudizi degli N giudici.

Si scriva un programma completo in linguaggio C per determinare il vincitore, ovvero il nuotatore che ha collezionato il punteggio più alto. Si supponga di utilizzare una matrice 100x10 (100 righe, una per nuotatore e 10 colonne, una per giudice) per memorizzare il giudizio di ciascun giudice su ciascun nuotatore.

In particolare, si scriva:

A) L'intestazione del programma, dichiarando opportunamente tutti i tipi e le variabili necessarie.

B) L'acquisizione da tastiera del numero di giudici e del numero di nuotatori (controllando che entrambi i numeri siano positivi e minori o uguali del numero massimo previsto) nella specifica gara, nonché del giudizio di ogni giudice su ogni nuotatore in tale gara.

C) Il calcolo e l'immagazzinamento in un'opportuna struttura dati del punteggio complessivo di ogni nuotatore.

D) Il calcolo e la stampa a video del punteggio e del numero del nuotatore vincitore (si supponga che ogni nuotatore sia identificato dal numero di riga nella matrice dei giudizi in cui si trovano memorizzati i suoi punteggi).

E) Si riprogettino infine le strutture dati del programma in modo da avere un tipo di struct *Nuotatore* contenente il nome e il numero del nuotatore e tutti i giudizi che gli sono stati attribuiti dai giudici (al più 10 come indicato sopra), e si dichiari un array di *Nuotatore* (al più 100 come indicato sopra). Si mostri come si dovrebbe modificare la parte di programma al punto C per consentire l'utilizzo di queste nuove strutture dati. Per rispondere a questa domanda, si dichiarino tutte le variabili necessarie e si supponga che queste siano state già opportunamente inizializzate.

Soluzione:

```
// A)
#include <stdio.h>
#include <stdlib.h>

#define MAXN 10
#define MAXK 100

int main() {
	int K, N;
	int giudizi[MAXK][MAXN];
	float tot[MAXK], max;
	int i, j;
	int posmax;

// B)
	printf("Quanti giudici ci sono? ");
	scanf("%d", &N);
	while(N <= 0) {
		printf("\nIl numero deve essere positivo\n");
```

```
            printf("Quanti giudici ci sono?");
            scanf("%d", &N);
    }

    printf("Quanti nuotatori ci sono?");
    scanf("%d", &K) ;
    while(K <= 0) {
            printf("\nIl numero deve essere positivo\n");
            printf("Quanti nuotatori ci sono? ");
            scanf("%d", &K);
    }

    for (i = 0; i < K; i++) {
            printf("Nuotatore %d:\n", i + 1);
            for (j = 0; j < N; j++) {
                    printf("Inserire giudizio giudice %d (valore 0 o 1): ", j + 1);
                    scanf("%d", &giudizi[i][j]);
                    while(giudizi[i][j] != 0 && giudizi[i][j] != 1) {
                    printf("\nIl giudizio deve essere un valore 0 o 1\n");
                            printf("Inserire giudizio giudice %d (valore 0 o 1): ", j + 1);
                    scanf("%d", &giudizi[i][j]);
                    }
            }
    }
//C)
    for (i = 0; i < K; i++) {
            tot[i] = 0;

            for (j = 0; j < N; j++) {

                    tot[i] = tot[i] + giudizi[i][j];
            }

            tot[i] = tot[i] / N;
    }

// D)
    max = tot[0];
    posmax = 0;
    for (i = 1; i < K; i++) {
            if (tot[i] > max) {
                    max = tot[i];
                    posmax = i ;
            }
    }
    printf("Il vincitore e' il nuotatore numero %d, con il punteggio %3.2f\n", posmax + 1, max);

    return 0;
}

// E)
```

```
typedef char stringa[30];
typedef struct {
  stringa nome;
  int giudizi[MAXN];
} nuotatore;
typedef nuotatore nuotatori[MAXK];


nuotatori arrayN;

for (i = 0; i < K; i++) {
    tot[i] = 0;
    for (j = 0; j < N; j++) {
        tot[i] = tot[i] + arrayN[i].giudizi[j];
    }
    tot[i] = tot[i] / N;
}
```

Esercizio 2 (10 punti)

Un'agenzia di trading online vuole memorizzare l'andamento del valore dei titoli che controlla. La memorizzazione viene effettuata in **500** istanti temporali equidistanti. I dati vengono salvati nel file MATLAB **log.mat** che contiene:

- la matrice **titoli**, le cui righe rappresentano i diversi titoli controllati e le cui colonne rappresentano i vari istanti in cui sono stati memorizzati i valori di tali titoli (quindi ogni cella della matrice contiene il valore di un titolo in un dato istante)
- il vettore colonna **andamento**, con lo stesso numero di righe della matrice **titoli**, che contiene un valore numerico per ogni titolo, indicativo del suo andamento complessivo crescente o decrescente

1. Scrivere in linguaggio MATLAB una funzione **splittaMatrice** che:
   - riceva in input una matrice **titoliTot** (con la stessa struttura di **titoli**), un vettore **andamentoTot** (con stessa struttura di vettore **andamento**) e uno scalare **soglia**;
   - fornisca in output due matrici **titoliOver** e **titoliUnder** (ognuna con la stessa struttura di **titoliTot**). **titoliOver** include solo le righe di **titoliTot** corrispondenti agli elementi di **andamentoTot** con valore maggiore o uguale di **soglia**. **titoliUnder**, invece, include le righe di **titoliTot** corrispondenti agli elementi di **andamentoTot** con valore minori di **soglia**.

2. Scrivere in linguaggio MATLAB uno script che:
   A) legga dal file **log.mat** i due dati memorizzati: **titoli** e **andamento**
   B) richiami la funzione **splittaMatrice** per separare **titoli** nelle due matrici **titoliOver** e **titoliUnder**, per un valore di **soglia** pari a 0
   C) crei un vettore **x** che contenga i 500 istanti di memorizzazione
   D) disegni su due grafici separati (che includano il titolo del grafico e il nome dei due assi) l'andamento dei titoli in **titoliOver** e **titoliUnder**, in funzione di **x**.

Soluzione:

1.
```
function [titoliOver, titoliUnder] = splittaMatrice(titoliTot, andamentoTot, soglia)

% Riceve in input la matrice titoliTot, il vettore andamentoTot e uno scalare soglia.
% Fornisce in output due matrici: titoliOver e titoliUnder
% - titoliOver include le righe di titoliTot corrispondenti agli elementi di andamentoTot
%   con valore maggiore o uguale di soglia
% - titoliUnder include le righe di titoliTot corrispondenti agli elementi di andamentoTot
%    con valore minore di soglia.

titoliOver  = titoliTot(andamentoTot >= soglia, :);
titoliUnder  = titoliTot(andamentoTot < soglia, :);
```

2.
```
close all; clear all; clc;

% A)
load('log.mat');

% B)
soglia = 0;
[titoliOver, titoliUnder] = splittaMatrice(titoli, andamento, soglia);
```

```
% C)
x = 1: 500;


% D)
figure
ylabel('Valore titolo');
xlabel('Tempo');
title('Titoli positivi');
if titoliOver
   [r, c] = size(titoliOver);
   hold on;
   for ii=1:r
      plot(x, titoliOver(ii, :));
   end
   hold off;
end

figure
ylabel('Valore titolo');
xlabel('Tempo');
title('Titoli in decrecita');
if titoliUnder
   [r, c] = size(titoliUnder);
   hold on;
   for ii=1:r
      plot(x, titoliUnder(ii, :));
   end
   hold off;
end
```

Esercizio 3 (6 punti)

Si assuma di avere un sistema operativo con un quanto di tempo di 25 ms e un processo P1 che riesce a decodificare 80 byte di segnale audio per ogni millisecondo di esecuzione.

Supponendo che una corretta riproduzione dell'audio richieda che vengano decodificati almeno 128 kbit/s, si risponda alle seguenti domande giustificando le risposte:

1. Se il processo P1 è l'unico attivo, può rispettare la velocità di codifica richiesta?
2. Supponiamo di attivare contemporaneamente a P1 altri *3* processi che abbiano lo stesso quanto di tempo e la stessa priorità di P1 e che ciascun cambio di contesto fra processi implichi un ritardo di 1 ms. Il processo P1 riuscirà a rispettare il limite richiesto per la corretta riproduzione?

Soluzione:

1. Se P1 è l'unico attivo, avrà l'intera CPU per se, quindi potrà decodificare 80 * 8 * 1000 bit/s = 640 kbit/s; quindi può rispettare la velocità di codifica richiesta.
2. Se sono attivi altri *n* processi, P1 non avrà l'intero tempo a disposizione, ma solo una parte di esso, sottraendo i quanti di tempo degli altri *n* processi e gli *n* cambi di contesto da 1 ms. La porzione di tempo che P1 avrà per sé sarà quindi:

   25 / (25 * (1 + n) + n) = 25 / (26 * n + 25)

   Se calcoliamo quanti bit riesce a codificare in questa porzione di tempo e lo confrontiamo con il limite avremo:

1000 * 80 * 8 * 25 / (26 * n + 25) > 128 k
640 k * 25 / (26 * n + 25) > 128 k
25 * 5 > 26 * n + 25
100 > 26 * n
n < 3.8 ovvero 4

Se n = 3, il processo riuscirà quindi a decodificare rispettando il limite stabilito.

Esercizio 1 (10 punti)

Una società che organizza eventi vuole sapere quale degli **E** eventi organizzati (al più 50) è piaciuto di più. A questo scopo raccoglie le valutazioni di un gruppo di **S** spettatori (al più 200) che ha partecipato a tutti gli eventi. Ognuno di tali spettatori valuta ogni evento con un valore numerico intero **1** (se ha apprezzato l'evento), o **0** (se non ha apprezzato l'evento); la valutazione complessiva di ogni evento (ognuno identificato da un numero progressivo) è un valore numerico tra 0 e 1 calcolato come media delle valutazioni degli **S** spettatori.

Si scriva un programma completo in linguaggio C per determinare l'evento più apprezzato, ovvero l'evento che ha ottenuto la valutazione complessiva più alta. Si supponga di utilizzare una matrice 200x50 (200 righe, una per spettatore e 50 colonne, una per evento) per memorizzare il giudizio di ciascuno spettatore su ciascun evento.

In particolare, si scriva:

A) L'intestazione del programma, dichiarando opportunamente tutti i tipi e le variabili necessarie.

B) L'acquisizione da tastiera del numero di eventi e del numero di spettatori valutanti (controllando che entrambi i numeri siano positivi e minori o uguali del numero massimo previsto), nonché della valutazione di ogni evento da parte di ognuno di tali spettatori.

C) Il calcolo e l'immagazzinamento in un'opportuna struttura dati della valutazione complessiva di ogni evento.

D) Il calcolo e la stampa a video della valutazione complessiva e del numero dell'evento più apprezzato (si supponga che ogni evento sia identificato dal numero di colonna nella matrice dei giudizi in cui si trovano memorizzati i giudizi attribuiti ad esso).

E) Si riprogettino infine le strutture dati del programma in modo da avere un tipo di struct *Evento* contenente il nome dell'evento e tutti i giudizi che sono stati attribuiti a tale evento dagli spettatori valutanti (al più 200 come indicato sopra), e si dichiari un array di *Evento* (al più 50 come indicato sopra). Si mostri come si dovrebbe modificare la parte di programma al punto C per consentire l'utilizzo di queste nuove strutture dati. Per rispondere a questa domanda, si dichiarino tutte le variabili necessarie e si supponga che queste siano state già opportunamente inizializzate.

Soluzione:

```c
// A)
#include <stdio.h>
#include <stdlib.h>

#define MAXS 200
#define MAXE 50

int main() {
        int E, S;
        int valutazioni[MAXE][MAXS];
        float tot[MAXE], max;
        int i, j;
        int posmax;
```

```
// B)
        printf("Quanti spettatori valutanti ci sono?");
        scanf("%d", &S);
        while(S <= 0 || S > MAXS) {
                printf("\nIl numero deve essere positivo e minore o uguale di %d\n", MAXS);
                printf("Quanti spettatori valutanti ci sono?");
                scanf("%d", &S);
        }

        printf("Quanti eventi si valutano? ");
        scanf("%d", &E) ;
        while(E <= 0 || E > MAXE) {
                printf("\nIl numero deve essere positivo e minore o uguale di %d\n", MAXE);
                printf("Quanti eventi si valutano? ");
                scanf("%d", &E);
        }

for (i = 0; i < E; i++) {
        printf("Evento %d:\n", i + 1);
        for (j = 0; j < S; j++) {
                        printf("Spettatore %d:\n ", j + 1);
                        printf("Inserire valutazione (valore 0 o 1): ");
                scanf("%d", &valutazioni[i][j]);
                        while(valutazioni[i][j] != 0 && valutazioni[i][j] != 1) {
                        printf("\nIl valore della valutazione deve essere 0 o 1\n");
                                printf("Inserire valutazione (valore 0 o 1): ");
                        scanf("%d", &valutazioni[i][j]);
                }
        }
        }
//C)
        for (i = 0; i < E; i++) {
                tot[i] = 0;
                for (j = 0; j < S; j++) {
                        tot[i] = tot[i] + valutazioni[i][j];
                }

                tot[i] = tot[i] / S;
        }

// D)
        max = tot[0];
        posmax = 0;
        for (i = 1; i < E; i++) {
                if (tot[i] > max) {
                        max = tot[i];
                        posmax = i ;
                }
        }
        printf("L'evento piu' apprezzato e' il numero %d, con la valutazione %3.2f\n", posmax + 1, max);
```

```
                return 0;
        }
// E)
typedef char stringa[30];
typedef struct {
  stringa nome;
  int giudizi[MAXS];
} evento;
typedef evento eventi[MAXE];


eventi arrayE;


for (i = 0; i < E; i++) {
    tot[i] = 0;
    for (j = 0; j < S; j++) {
        tot[i] = tot[i] + arrayE[i].giudizi[j];
    }
    tot[i] = tot[i] / S;
}
```

Esercizio 2 (10 punti)

Un comitato sportivo di sci vuole memorizzare le performance degli sportivi che percorrono la più ripida delle piste del comprensorio. Per ogni atleta, la memorizzazione parte da quota *h0* = 3000 m e termina a quota *hend* = 1000 m, e viene effettuata in 100 punti diversi del tracciato equidistanti tra loro. I dati vengono salvati nel file **log.mat** che contiene:

- la matrice **atleti**, le cui righe rappresentano i diversi atleti monitorati, e le cui colonne rappresentano i vari tempi collezionati sul tracciato (quindi l'elemento (*i,j*) della matrice contiene il tempo in cui un atleta *i*-esimo è transitato per il *j*-esimo punto della pista);
- il vettore colonna **velocitaMedia**, con stesso numero di righe della matrice **atleti**, che contiene un valore numerico per ogni atleta, indicativo della sua velocità media lungo tutta la pista.

1. Scrivere in linguaggio Matlab una funzione **dividiMatrice** che:
   - riceva in input una matrice **atletiTot** (con la stessa struttura di **atleti**), un vettore **velocitaM** (con stessa struttura di vettore velocitaMedia) e uno scalare **soglia**;
   - fornisca in output due matrici **atletiOver** e **atletiUnder** (ognuna con la stessa struttura di **atletiTot**). **atletiOver** include solo le righe di **atletiTot** corrispondenti agli elementi di **velocitaM** con valore maggiore o uguale di **soglia**. **atletiUnder**, invece, include solo le righe di **atletiTot** corrispondenti agli elementi di **velocitaM** con valore minore di **soglia**.

2. Scrivere in linguaggio Matlab uno script che:
   A) legga dal file **log.mat** i due dati memorizzati: **atleti** e **velocitaMedia**;
   B) richiami la funzione **dividiMatrice** per separare **atleti** nelle due matrici **atletiOver** e **atletiUnder**, per un valore di **soglia** pari a 10;
   C) crei un vettore **x** che contenga i 100 punti di memorizzazione;
   D) disegni su due grafici separati (che includano il titolo del grafico e il nome dei due assi) l'andamento degli atleti con velocità media sopra o pari al valore soglia 10 e di quelli sotto tale soglia.

Soluzione:

1.
```
function [atletiOver, atletiUnder] = dividiMatrice(atletiTot, velocitaM, soglia)

% Riceve in input la matrice atletiTot, il vettore velocitaM e uno scalare soglia.
% Fornisce in output due matrici: atletiOver e atletiUnder
% - atletiOver include le righe di atletiTot corrispondenti agli elementi di velocitaM
%   con valore maggiore o uguale di soglia
% - titoliUnder include le righe di atletiTot corrispondenti agli elementi di velocitaM
%    con valore minore di soglia.

atletiOver = atletiTot(velocitaM >= soglia, :);
atletiUnder  = atletiTot(velocitaM < soglia, :);
```

2.
```
close all; clear all; clc;

% A)
load('log.mat');

% B)
soglia = 10;
```

```
        [atletiOver, atletiUnder] = dividiMatrice(atleti, velocitaMedia, soglia);

        % C)
        x = 1: 100;

        % D)
        figure
        ylabel('Velocità medie');
        xlabel('Tempo');
        title('Atleti sopra o pari a soglia');
if atletiOver
   [r, c] = size(atletiOver);
   hold on;
   for ii=1:r
      plot(x, atletiOver(ii, :));
   end
   hold off;
end

        figure
        ylabel('Velocità medie');
        xlabel('Tempo');
        title('Atleti sotto soglia');
if atletiUnder
   [r, c] = size(atletiUnder);
   hold on;
   for ii=1:r
      plot(x, atletiUnder(ii, :));
   end
   hold off;
end
```

Esercizio 3 (6 punti)

Si assuma di avere un sistema operativo con un quanto di tempo di 50 ms e un processo P1 che riesce a decodificare 60 byte di segnale audio per ogni millisecondo di esecuzione.

Supponendo che una corretta riproduzione dell'audio richieda che vengano decodificati almeno 128 kbit/s, si risponda alle seguenti domande giustificando le risposte:

1. Se il processo P1 è l'unico attivo, può rispettare la velocità di codifica richiesta?
2. Supponiamo di far girare contemporaneamente a P1 altri *4* processi che abbiano lo stesso quanto di tempo e la stessa priorità di P1 e che ciascun cambio di contesto fra processi implichi un ritardo di 1 ms; il processo P1 riuscirà a rispettare il limite richiesto per la corretta riproduzione?

Soluzione:

1. Se P1 è l'unico attivo, avrà l'intera CPU per se, quindi potrà decodificare 60 * 8 * 1000 bit/s = 480 kbit/s; quindi può rispettare la velocità di codifica richiesta.
2. Se sono attivi altri *n* processi, P1 non avrà l'intero tempo a disposizione, ma solo una parte di esso, sottraendo i quanti di tempo degli altri *n* processi e gli *n* cambi di contesto da 1 ms. La porzione di tempo che P1 avrà per sé sarà quindi:

   50 / (50 * (1 + n) + n) = 50 / (51 * n + 50)

   Se calcoliamo quanti bit riesce a codificare in questa porzione di tempo e lo confrontiamo con il limite avremo:

1000 * 60 * 8 * 50 / (51 * n + 50) > 128 k
480 k * 50 / (51 * n + 50) > 128 k
3.75 * 50 > (51 * n + 50)
2.75 * 50 / 51 > n
n < 2.69 ovvero 3

Siccome n = 4, il processo P1 non rispetterà il limite richiesto.

.