	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 3 Febbraio 2009		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA

- Il presente plico contiene 4 esercizi, deve essere debitamente compilato con cognome e nome, numero di matricola, posizione durante lo scritto (comunicata dal docente).
- 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, purché in modo ben marcato e leggibile**.
- È **vietato** utilizzare **calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà **annullata** la sua prova.
- È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.
- 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.
- Gli studenti degli anni precedenti (e solo loro) *possono* svolgere per gli esercizi 1 e 2 le versioni in C.

Esercizio 1 (6 punti)
Scrivere una funzione che prende come parametro due matrici A e B, della stessa dimensione, e restituisce i seguenti 3 valori:

1) Il numero di elementi uguali in posizioni corrispondenti.
2) Il numero di elementi di A *massimali* per la matrice B, cioè il numero di posizioni, nella matrice A, in cui è presente un elemento che è maggiore o uguale a tutti gli elementi presenti nella matrice B.
3) il numero massimo di elementi di A massimali per la matrice B (secondo la definizione precedente) presenti in una stessa colonna della matrice A.

Per esempio, se A e B sono definite come segue:
A = [4 11 3; 3 8 5; 0 10 2; 6 3 9];
B = [1 2 3; 3 4 5; 1 1 3; 6 3 7];
Il numero di elementi uguali in posizioni corrispondenti è 5: l'elemento di posizione (1, 3); gli elementi di posizione (2, 1) e (2, 3); gli elementi di posizione (4, 1) e (4, 2).
Il numero di elementi di A massimali per B è pari a 4, infatti gli elementi 11, 8, 10 e 9 sono gli elementi di A maggiori di tutti gli elementi in B.
Il numero massimo di elementi di A massimali per B che si trovano sulla stessa colonna di A è pari a 3, infatti nella seconda colonna di A si trovano i tre valori 11, 8 e 10, mentre nella terza colonna si trova solo il valore 9.

Per sviluppare la funzione è possibile utilizzare le funzioni descritte nella Tabella 1. È inoltre possibile utilizzare altre funzioni di libreria disponibili in Matlab/Octave o funzioni definite al momento, purchè venga spiegato chiaramente quale è lo scopo della funzione usata, che cosa riceve come parametri di ingresso e cosa produce in uscita.

Nota per gli studenti degli anni precedenti: Chi ha seguito negli anni precedenti può svolgere questo esercizio in C (ed appoggiarsi ad altre funzioni C già sviluppate o da sviluppare ma delle quali si spiegherà lo scopo e l'intestazione). Per capire l'esempio si tenga presente che in Matlab/Octave gli indici di riga e colonna delle matrici partono dal valore 1 e che nell'assegnamento di valore alle matrici ciascun elemento è separato dagli altri tramite uno spazio (o una virgola) e le righe sono separate da ";".

Tabella 1

max()
Sintassi
`max(A)`
Descrizione
Se `A` è un vettore, `max(A)` restituisce l'elemento più grande in `A`
Se `A` è una matrice, `max(A)` restituisce un vettore riga che contiene gli elementi più grandi di ciascuna colonna della matrice `A`
Esempio
`max([1,5;3,4;5,2])` restituisce `[5,5]`
	min()
	Sintassi
	`min(A)`
	Descrizione
	Se `A` è un vettore, `min(A)` restituisce l'elemento più piccolo in `A`
	Se `A` è una matrice, `min(A)` restituisce un vettore riga che contiene gli elementi più piccoli di ciascuna colonna della matrice `A`
	Esempio
	`min([1,5;3,4;5,2])` restituisce `[1,2]`

any()
Sintassi
`any(A)`
Descrizione
Se `A` è un vettore, `any(A)` restituisce `1` (`true`) se `A` contiene almeno un valore numerico diverso da `0` o un `1` (`true`), altrimenti restituisce `0` (`false`)
Se `A` è una matrice, `any(A)` restituisce un vettore riga logico in cui ciascun elemento è un `1` (`true`) se la corrispondente colonna di `A` contenga almeno un valore numerico diverso da `0` o un `1` (`true`).
Esempio
`any([0,1,0;0,0,1])` restituisce `[0,1,1]`
	all()
	Sintassi
	`all(A)`
	Descrizione
	Se `A` è un vettore, `all(A)` restituisce `1` (`true`) se `A` contiene solo valori numerici diversi da `0` o solo `1` (`true`), altrimenti restituisce `0` (`false`)
	Se `A` è una matrice, `all(A)` restituisce un vettore riga logico in cui ciascun elemento è un `1` (`true`) se la corrispondente colonna di `A` contenga solo valori numerici diversi da `0` o solo `1` (`true`).
	Esempio
	`all([0,1,0;0,1,1])` restituisce `[0,1,0]`

sum()
Sintassi
`sum(A)`
Descrizione
Se `A` è un vettore, `sum(A)` restituisce la somma degli elementi in `A`
Se `A` è una matrice, `sum(A)` restituisce un vettore riga contente la somma degli elementi di ciascuna colonna di `A`
Esempio
`sum([1,5,7;3,4,2])` restituisce `[4,9,9]`
	size()
	Sintassi
	`size(A)`
	Descrizione
	Se A è un vettore, `size(A)` restituisce la lunghezza del vettore
	Se A è una matrice, `size(A)` restituisce un vettore riga contente il numero di righe e di colonne della matrice `A`
	Esempio
	`size([1,5,7;3,4,2])` restituisce `[2,3]`

Soluzione

Versione Matlab/Octave
Nota: riportiamo una soluzione compatta. Altre soluzioni meno compatte ma corrette sono ritenute accettabili.
Osserviamo che un elemento della matrice A è maggiore o uguale a tutti gli elementi della matrice B se e solo se è maggiore o uguale al massimo fra di essi. Quindi per verificare che un elemento di A sia massimale per B è sufficiente confrontarlo con l'elemento massimo di B, calcolato preventivamente e *una tantum*.

```
function [r1,r2,r3]=soluzioneEs1a(A, B)
t1 = A==B;
M = max(max(B));
t2 = A >= M;

r1 = sum(sum(t1));
r2 = sum(sum(t2));
r3 = max(sum(t2));
```

Versione C

```
#define N 20

typedef enum {false, true} boolean;
```

```
/* massimale restituisce vero se il valore x e` maggiore di tutti i valori in matr. nrighe ed ncolonne
indicano il numero di righe e colonne di matr che sono occupate con valori significativi */
boolean massimale(int x, int matr[][N], int nrighe, int ncolonne);

void cm1(int A[][N], int B[][N], int nrighe, int ncolonne, int *r1, int *r2, int *r3)
{
   int i, j, massimaliCol;
   int uguali = 0, massimali = 0, maxMassim = 0;
   for (j=0; j<ncolonne; j++)
   {
     massimaliCol = 0
     for (i=0; i<nrighe; i++)
     {
       if(A[i][j]==B[i][j])
         uguali = uguali + 1;
       if(massimale(A[i][j], B, nrighe, ncolonne)==true)
         massimaliCol = massimaliCol+1;
     }
     if(massimaliCol > maxMassim) maxMassim = massimaliCol;
     massimali = massimali + massimaliCol;
   }
   /* preparazione dei valori che vengono prodotti per il chiamante attraverso il passaggio di
      parametri per indirizzo */
   *r1 = uguali;
   *r2 = massimali;
   *r3 = maxMassim;
}
```

Esercizio 2 (3 punti)
Facendo uso della funzione di ordine superiore acc spiegata a lezione (e di qui si riporta il codice qui sotto) codificare la funzione modulo(v) che prende come argomento un vettore di numeri $v=[v_1, v_2, \ldots v_n]$ e calcola e restituisce come risultato il valore $\prod_{i=1}^{n} v_i * \sum_{i=1}^{n} v_i$

Codice della funzione acc.

```
function [x]=acc(f, a, u)
        x = u;
        for i=1:length(a)
                x = f(x, a(i));
        end
```

Soluzione

```
function ris = modulo(v)
  prod = @(x, y) x*y;
  sum = @(x, y) x+y;
  ris = acc(prod, v, 1)*acc(sum, v, 0);
```

Esercizio 2 solo per gli studenti degli anni precedenti (3 punti)
Si consideri una lista dinamica di interi strettamente positivi, i cui elementi sono del tipo ELEMENTO definito come:

```
 typedef struct El {
   int dato;
   struct El *next;
 } ELEMENTO;
typedef ELEMENTO *lista;
```

Si codifichi in C una funzione somma avente il seguente prototipo:
  int somma(lista Testa, int M)
Tale funzione riceve come parametro la testa della lista e un intero M e restituisce la somma dei soli valori della lista che sono minori di M.

Soluzione

```
/* versione ricorsiva */
int somma(lista Testa, int M)
{
   if(Testa == NULL) return 0;
   else if(Testa->dato < M)
      return Testa->dato+prodotto(Testa->next, M);
   else return prodotto(Testa->next, M);
}

/* versione iterativa */
int somma(lista Testa, int M)
{
  int s=0;
  while(Testa!=NULL)
  {
      if(Testa->dato < M)
        s = s+Testa->dato;
      Testa = Testa->next;
  }
  return s;
}
```

Esercizio 3 (4.5 punti)

Si considerino i due seguenti numeri X e Y, codificati secondo lo standard virgola mobile IEEE visto a lezione.

X s = 1
m = 00000000000000000000000
c = 01111111
Y s = 1
m = 01001001010101010101010
c = 01011001

Si calcoli, nel modo più semplice possibile (cioè facendo un minimo di calcoli aritmetici) la rappresentazione del loro prodotto X×Y

Soluzione

Poiché X ha mantissa composta da soli zeri, il suo valore è una potenza di 2; la sua caratteristica vale 127 (7 uni di fila valgono 2^7-1=127) quindi codifica l'esponente 0 (=127-127); perciò il valore di X è esattamente -1 (per via del bit di segno 1); in conclusione $X \times Y = -Y$, e la rappresentazione di -Y è la stessa di Y, eccetto per il bit di segno

-Y s = 0
m = 01001001010101010101010
c = 01011001

Esercizio 4 (3.5 punti)

Un sistema dispone di una memoria fisica indirizzabile con indirizzi a 11 bit; inoltre è dotato di memoria virtuale con paginazione caratterizzata dai seguenti parametri: l'indirizzo virtuale è di 12 bit e le pagine sono di 1Kbyte.

Rispondere alle seguenti domande giustificando le risposte:

a) Qual è la dimensione della memoria virtuale e della memoria fisica indirizzabile?
b) Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono

Soluzione

a) la memoria virtuale indirizzabile è pari a 4 Kbyte (2^12 byte), la memoria fisica indirizzabile è pari a 2 Kbyte (2^11 byte).
b) Indirizzo Virtuale: 12 bit NPV: 2 bit offset virtuale: 10 bit
   Indirizzo Fisico: 11 bit NPF: 1 bit offset fisico: 10 bit