	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Appello del 7 Febbraio 2011		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 4 esercizi e 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 e 30 minuti.
- Non separate questi fogli. Scrivete la soluzione **solosui 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 propria 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.

Esercizio 1 (8 punti)

L'accesso dei veicoli in una zona urbana è regolamentato come segue. I veicoli sono divisi in **tre classi di inquinamento** in base al **tipo di motore** e al **tipo di trasporto** a cui sono adibiti:

- Classe 0: veicoli **gpl** adibiti al trasporto di **persone**
- Classe 1: veicoli **benzina** e **diesel** adibiti al trasporto di **persone**; veicoli **gpl** adibiti al trasporto **merci**
- Classe 2: i rimanenti

Per ogni classe di veicolo sono definite tre tariffe di ingresso: **giornaliero, mensile** ed **annuale**.

Sulla base delle seguenti dichiarazioni:

```
typedef enum{benzina,diesel,gpl} motore;
typedef enum{persone,merci} tipo;
typedef enum{giornaliero,mensile,annuale} ingresso;

typedef struct {
     char targa[7]; //targa del veicolo
     motore m; // motore del veicolo
     tipo t;   // tipologia di trasporto
     ingresso i; // ingresso richiesto dal veicolo
}veicolo;

/*La seguente e' la matrice con le tariffe di ingresso, dove l'indice
di riga corrisponde alla classe di inquinamento del veicolo e
l'indice di colonna identifica il tipo di ingresso (0=giornaliero,
1=mensile, 2=annuale) */

float tariffe[3][3];

veicolo v[N];
veicolo vC0[N];
float fatture[N];
```

Scrivere un frammento di programma in C che:

A) calcoli per ciascun veicolo presente all'interno dell'array **v** (che si assuma già inizializzato) il costo dell'ingresso alla zona urbana in base ai costi specificati nella matrice **tariffe** (che si assuma già inizializzata) e salvi tale valore all'interno dell'array **fatture**;

B) memorizzi nell'array **vC0**, senza lasciare spazi, tutti i veicoli contenuti in **v** che appartengono alla classe 0.

Note. Riportare soltanto il frammento di programma richiesto dichiarando le eventuali variabili aggiuntive necessarie.

Soluzione (A e B insieme)

```
int classe;
int k = 0;
int i;
for (i=0; i<N; i++) {
     if (v[i].m == gpl && v[i].t == persone)
          classe = 0;
     else if ( (v[i].m == gpl && v[i].t == merci) ||
               (v[i].m != gpl && v[i].t == persone) )
          classe = 1;
     else
          classe = 2;
     fatture[i] = tariffe[classe][v[i].i];
     if (classe == 0)    {
          vC0[k] = v[i];
          k++;
     }
}
```

Esercizio 2 (10 punti)

Un metodo per calcolare il valore approssimato della radice quadrata di un numero reale non negativo z, detto metodo babilonese, utilizza la seguente relazione ricorsiva:

$$\begin{cases} x(n) = \frac{1}{2}(x(n-1) + \frac{z}{x(n-1)}) \\ x(0) = 1 \end{cases}$$

Dove *n* è un intero non-negativo arbitrario, mentre *x*(*n*) rappresenta un'approssimazione della radice quadrata del numero *z*. Per esempio, se z è pari a 2 e n è pari a 3 il valore di x(n) ottenuto applicando la formula indicata sopra è 1.4142 (che è una buona approssimazione della radice di 2).

L'errore di approssimazione associato a *x*(*n*) è definito come segue:

$$errore(n) = |x(n) - x(n-1)|$$

e decresce al crescere di *n*. Per esempio, se si calcola la radice di 2 con *n* pari a 1 si ottiene il valore 1.5000 con errore 0.5000. Se invece si calcola la radice dello stesso valore con *n* pari a 2 si ottiene il valore 1.4167 con errore 0.0833.

1. Si scriva una funzione ricorsiva *sqrt1*(*z*,*n*) per MATLAB/Octave che restituisca un'approssimazione della radice quadrata di *z* e il relativo errore utilizzando il metodo babilonese. Quando *n* è pari a zero, si assuma un valore dell'errore pari a *inf*.
2. Si scriva una *funzione sqrt2*(*z*,*err*) per MATLAB/Octave che, utilizzando la funzione *sqrt1*, restituisca un valore approssimato della radice quadrata di *z* con errore non superiore al valore *err* fornito come parametro.

Nella soluzione di entrambi i quesiti non è permesso utilizzare alcuna funzione di libreria di MATLAB/Octave a parte la funzione *abs(x)* per il calcolo del valore assoluto di *x*.

Soluzione

```
function [r err] = sqrt1(z,n)
  if (n == 0) % caso base
      r=1;
      err = inf;
  elseif (n > 0) % caso ricorsivo
      [tmp_r tmp_err] = sqrt1(z,n-1);
      r = 0.5 * (tmp_r + z / tmp_r);
      err = abs(r - tmp_r);
  end

function r = sqrt2(z,err)
  k = 0;
  tmp_err = inf;
  while (tmp_err > err)
      k = k + 1;
      [r tmp_err] = sqrt1(z,k);
  end
```

Esercizio 3 (10 punti)
Si consideri il famoso problema delle 8 regine, che fu pubblicato per la prima volta su una rivista di scacchi tedesca nel 1848 e alla cui soluzione si dedicò anche Gauss.

Il problema consiste nel trovare una disposizione delle otto regine su una scacchiera 8x8 in modo tale che nessuna di esse sia minacciata dalle altre. In altre parole, dato che la regina può spostarsi in orizzontale, in verticale e in diagonale di un qualsiasi numero di caselle, ogni regina deve avere la sua riga, la sua colonna e le sue due diagonali libere. La figura qui sotto mostra una possibile configurazione (le R rappresentano le regine sulla scacchiera).

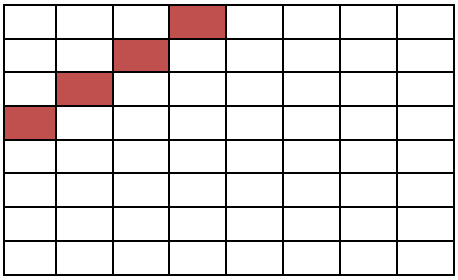
			R				
						R	
		R					
							R
	R						
				R			
R							
					R		

Quesito 1: Si sviluppi una funzione MATLAB di nome **ottoRegine** che riceve come parametro una matrice 8x8, che rappresenta la scacchiera su cui sono disposte le 8 regine, e restituisce ***true*** se la configurazione delle 8 regine è corretta, ***false*** altrimenti. Si supponga che la matrice contenga il valore 0 in tutte le posizioni libere e il valore 1 nelle posizioni occupate dalle regine. Per costruire tale funzione, si utilizzino le funzioni **estraiDiagonale(a,r,c)** ed **estraiAntiDiagonale(a,r,c)** che prendono come parametro una matrice **a**, un numero di riga **r** e un numero di colonna **c**.

estraiDiagonale restituisce, in un vettore, tutti gli elementi che si trovano sulla diagonale di **a** che ha origine nell'elemento di posizione **(r,c)**.

estraiAntiDiagonale restituisce, in un vettore, tutti gli elementi che si trovano sull'antidiagonale (o diagonale secondaria) di **a** che hanno origine nell'elemento di posizione **(r,c).**

Si noti che le origini di tutte le diagonali si trovano nella prima riga e nella prima colonna della matrice, e che le origini di tutte le antidiagonali si trovano nella prima riga e nell'ultima colonna. L'esempio seguente mostra l'antidiagonale che ha origine nella posizione (1, 4).



Soluzione

```
function [ris] = ottoRegine(a)
ris = true;
[righe, colonne] = size(a);
% controlla che ci sia una regina per colonna

if ~sum(a) ris = false;
    % controlla che ci sia una regina per riga
elseif ~sum(a, 2) ris = false;
    % controlla le diagonali
else
    % tutte le diagonali che passano per gli elementi della prima colonna
    % tutte le antidiagonali passano per gli elementi dell'ultima colonna

    for r = 1 : righe
        if(sum(estraiDiagonale(a, r, 1) ~= 0) > 1)
            ris = false;
            return;
        end
        if(sum(estraiAntiDiagonale(a, r, colonne) ~= 0) > 1)
            ris = false;
            return;
        end
    end
    % tutte le diagonali passano per gli elementi della prima riga
    % tutte le antidiagonali passano per gli elementi della prima riga

    for c = 1 : colonne
        if(sum(estraiDiagonale(a, 1, c) ~= 0) > 1)
            ris = false;
            return;
        end
```

```
        if(sum(estraiAntiDiagonale(a, 1, c) ~= 0) > 1)
            ris = false;
            return;
        end
    end
end

function vet = estraiDiagonale(a, rInit, cInit)

vet = [];
[righe, colonne] = size(a);

while(rInit <= righe & cInit <= colonne)
    vet = [vet, a(rInit, cInit)];
    rInit = rInit + 1;
    cInit = cInit + 1;
end

% funzione non richiesta dal testo ma utile per provare l'esempio
function vet = estraiAntiDiagonale(a, rInit, cInit)

vet = [];
[righe, colonne] = size(a);

while(rInit <= righe & cInit >= 1)
    vet = [vet, a(rInit, cInit)];
    rInit = rInit + 1;
    cInit = cInit - 1;
end
```

Esercizio 4 (6 punti)

1. Si determini la codifica del valore 0.125 secondo lo Standard IEEE 754-1985 a precisione singola, riportando i calcoli effettuati.
2. Si determini la codifica del valore 2.0 secondo lo stesso standard, sempre riportando i calcoli effettuati.
3. Tenendo presenti le risposte fornite ai precedenti punti, si consideri il seguente programma in C e si indichi l'effetto della sua esecuzione, motivando adeguatamente la risposta.

```
#include <stdio.h>
main() {
   float p, s; int i;

   p = 0.125; s = p;
   for (i=1; i<16; i++)
      s = s + p;
   printf("\nIl numero 0.125*16 ");
   if (s != 2.0) printf("non ");
   printf("e' uguale a 2.0");
}
```

Soluzione

1. Codifica di 0.125:
   Segno=0
   Esponente=01111100 (127-3)
   Significando= .00000000000000000000000

2. Codifica di 2.0:
   Segno=0
   Esponente=10000000 (127+1)
   Significando= .00000000000000000000000

Conti da riportare, da cui si vede che sia 0.125 sia 2.0, essendo entrambi potenze di due, vengono rappresentati in modo esatto.
Di conseguenza, non intervenendo errori di approssimazione, il programma stampa la scritta "`Il numero 0.125*16 e' uguale a 2.0`".

[si noti la similitudine e la differenza rispetto all'esercizio dato nella prima prova intermedia: dovrebbero essere favoriti gli studenti che hanno esaminato con attenzione e capito le soluzioni]