	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Appello 17 Luglio 2013		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 2 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola**.
- Il tempo a disposizione è di 1 ora.
- 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 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**.
- L'esame orale è parte integrante dell'esame e deve essere realizzato almeno sufficientemente per il superamento dell'esame complessivo.
- Per l'ammissione all'esame orale è necessario aver almeno impostato sufficientemente entrambi gli esercizi.

Esercizio 1

Il programma gestionale di un'azienda di trasporti è scritto in C ed utilizza le seguenti strutture dati

```
#define MAX_TRASPORTI 100
#define NUM_DIPENDENTI 10
typedef char stringa[100];
typedef enum{falso, vero} boolean;

typedef struct {
    stringa destinazione;
    boolean trasportoSpeciale; /* Un trasporto è speciale se si movimentano prodotti
                                  velenosi o molto pesanti o di grandi dimensioni */
    float numeroKm; /* Indica il numero di Km che è necessario percorrere per portare il
                       carico a destinazione e tornare indietro alla sede dell'azienda */
} Trasporto;

typedef struct{
    stringa nome;
    stringa cognome;
    int kmPerLitro; /*indica quanti km in media percorre con un litro di carburante*/
    int numTrasportiEffettuati;
    Trasporto listaTrasporti[MAX_TRASPORTI];
} Camionista;
```

Si assuma inoltre che nel main del programma siano state dichiarate le seguenti variabili

```
Camionista dipendenti[NUM_DIPENDENTI];
float litriConsumati[NUM_DIPENDENTI];
```

La variabile **dipendenti** registra i dati dei 10 camionisti dell'azienda. La variabile **litriConsumati** serve per tenere traccia del numero di litri di carburante consumati da ciascun camionista (il camionista nella posizione i-esima in **dipendenti** corrisponde al consumo di carburante indicato nella stessa posizione i-esima in **litriConsumati**).

Si risponda alle seguenti tre domande. Si tenga presente che queste sono completamente indipendenti l'una dall'altra. Nel caso quindi non si riesca a rispondere a una di esse, non si rinunci ad analizzare le successive.

1. Si assuma che la variabile **dipendenti** sia stata interamente popolata e si scriva un frammento di codice per scorrere **dipendenti** e per stimare il numero di litri di carburante consumati da ogni camionista. Tale valore deve quindi essere salvato nella posizione corrispondente del vettore **litriConsumati**. Il numero di litri consumati da un camionista si ottiene calcolando il numero totale di chilometri percorsi dal camionista nei suoi trasporti e dividendolo per il valore di kmPerLitro del camionista.
2. Si dichiari una seconda variabile **dipendentiNormali** e vi si copi, senza lasciare buchi, tutti i dipendenti che non hanno mai fatto un trasporto speciale (identificato dal valore dell'apposito campo della struttura)
3. Si assuma che il vettore **dipendentiNormali** sia stato popolato e che la variabile **numeroDipNormali** contenga il numero di elementi significativi di questo array. Si stampi a schermo il numero dei dipendenti che non hanno mai fatto un trasporto speciale, seguito dal nome e cognome di ciascuno di questi. Ad esempio:

   2 Dipendenti non hanno mai fatto trasporti speciali:
   Mario Rossi
   Paolo Bianchi

N.B. Se necessario, è possibile dichiarare variabili aggiuntive per svolgere i punti 1,2 e 3

SOLUZIONE

```
float totKm;
int flag, j, numeroDipNormali;

for(i = 0; i < NUM_DIPENDENTI; i++)
{
   totKm = 0;
   for(l = 0; l< dipendenti[i].numTrasportiEffettuati; l++)
   {
      totKm = dipendenti[i].listaTrasporti[l].numeroKm + totKm;
   }
   litriConsumati[i] = totKm/ dipendenti[i].kmPerLitro;
}

j=0;
for(i = 0; i < NUM_DIPENDENTI; i++)
{
   flag = 1;
   for(l = 0; l< dipendenti[i].numTrasportiEffettuati && flag == 1; l++)
   {
      if(dipendenti[i].listaTrasporti[l].trasportoSpeciale == vero)
         {
          flag = 0;
         }
   }
   if(flag == 1)
   {
      dipendentiNormali[j] = dipendenti[i];
      j++;
   }
}
numeroDipNormali = j;
printf("%d Dipendenti che non hanno mai fatto un trasporto speciale: \n", numeroDipNormali);
for(i = 0; i < numeroDipNormali; i++)
   printf("\n%s %s", dipendentiNormali[i].nome, dipendentiNormali[i].cognome);
}
```

Esercizio 2

Si consideri una versione semplificata della battaglia navale in cui le navi possono essere posizionate solo in orizzontale e ogni riga può contenere al massimo una nave.

Il campo di gioco di un singolo giocatore può essere rappresentato tramite la matrice **CampoGioco** di dimensione 5 x 5 in cui ogni cella della matrice può assumere solo il valore **0** o **1**. Il valore **0** rappresenta la presenza del mare e il valore **1** la presenza di un pezzo di nave. Le navi possono essere lunghe una, due, tre, quattro o cinque celle.
Ad esempio la seguente istanza della matrice **CampoGioco** rappresenta un campo di gioco in cui sono presenti 4 navi: una nave lunga 4 nella prima riga, una nave lunga 1 nella terza riga, una nave lunga 2 nella quarta riga e una nave lunga 4 nella quinta riga.

CampoGioco

0	**1**	**1**	**1**	**1**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**
1	**1**	**0**	**0**	**0**
1	**1**	**1**	**1**	**0**

Si realizzi uno script Matlab che:

- chiede all'utente di inserire il contenuto della matrice **CampoGioco**
- per ogni riga che contiene una nave visualizza a video il numero di riga e la lunghezza della nave presente al suo interno
- visualizza a video inoltre le seguenti statistiche:
    - il numero di navi presenti sul campo di gioco
    - la lunghezza della nave più corta presente sul campo di gioco
    - la lunghezza della nave più lunga presente sul campo di gioco
    - il numero di navi trovate per ogni lunghezza possibile.

Attenzione. I risultati a video devono essere visualizzati utilizzando un formato/stile analogo a quello riportato nel seguente esempio.

Esempio di output a video atteso per la matrice d'esempio **CampoGioco** riportata sopra.

La riga 1 contiene una nave lunga 4
La riga 3 contiene una nave lunga 1
La riga 4 contiene una nave lunga 2
La riga 5 contiene una nave lunga 4

Sono presenti 4 navi
Lunghezza nave più corta trovata: 1
Lunghezza nave più lunga trovata: 4
Numero di navi lunghe 1: 1
Numero di navi lunghe 2: 1
Numero di navi lunghe 3: 0
Numero di navi lunghe 4: 2
Numero di navi lunghe 5: 0

Soluzione

```
CampoGioco=input('Inserisci il campo di gioco: ');

% Calcolo (somma) per ogni riga il numero di 1 che contiene.
% Questo corrisponde alla lunghezza della nave in esso contenuta (0 = no
% nave)
lunghezzaNaviRighe=sum(CampoGioco,2);

% Trovo l'indice delle righe che contengono una nave
righeConNavi=find(lunghezzaNaviRighe~=0);

% Informazione sulle righe che contengono navi
for riga=righeConNavi'
    disp(['La riga ' num2str(riga) ' contiene una nave lunga ' num2str(lunghezzaNaviRighe(riga))]);
end

% Statistiche varie
disp(['Sono presenti ' num2str(size(righeConNavi,1)) ' navi']);

disp(['Lunghezza nave più corta trovata: ' num2str(min(lunghezzaNaviRighe(lunghezzaNaviRighe~=0)))]);

disp(['Lunghezza nave più lunga trovata: ' num2str(max(lunghezzaNaviRighe))]);


for lunghNave=1:5
disp(['Numero di navi lunghe ' num2str(lunghNave) ': ' num2str(sum(lunghezzaNaviRighe==lunghNave))]);
end
```