	Politecnico di Milano Facoltà di Ingegneria Industriale ***INFORMATICA B*** Prova in itinere del 2 Febbraio 2017		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA O CODICE PERSONA

- Il presente plico contiene **3 esercizi** e deve essere debitamente compilato con cognome e nome, numero di matricola e posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di **1 ora e 45 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** (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) assicurandosi comunque che **quanto scritto sia ben 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. **Non è tuttavia possibile consultare temi d'esame degli anni precedenti**.
- 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.
- Saranno valutate con un punteggio pieno solo le soluzioni che sfruttano le particolarità del linguaggio Matlab in particolare per la gestione di array e vettori

Esercizio 1A (8 punti)

Un'azienda ha installato un sistema che registra gli ingressi e le uscite dei propri dipendenti tramite l'uso di badge. All'entrata e all'uscita, ogni dipendente deve registrarsi passando il badge nel lettore posto all'ingresso. Siete stati incaricati di scrivere in Matlab parte del software per la gestione dei dati acquisiti.

Il programma Matlab è costituito da una serie di funzioni che lavorano sui seguenti dati:

1. Informazioni relative a ciascun ***dipendente***. Queste sono organizzate in una struct con i seguenti campi:
    - ***ID*** (un numero intero che identifica univocamente ogni dipendente),
    - ***nome*** (stringa)
    - ***cognome*** (stringa)
2. Il ***registro*** dei dipendenti. Questo è un array di struct che contiene in ogni suo elemento:
    - I campi di un dipendente (***ID, nome, cognome***).
    - La ***data*** che corrisponde a un ingresso e ad un'uscita del dipendente (si suppone che avvengano nello stesso giorno). Questa data è rappresentata con un numero intero di 8 cifre. Per esempio, il numero 20171223 corrisponde alla data 23 Dicembre 2017.
    - ***oraIngresso*** e ***oraUscita***, che vengono rappresentati con un numero intero che indica l'ora come il numero di minuti trascorsi dalla mezzanotte precedente (ad esempio, le ore 7:30 corrispondono a 450 minuti, infatti 7 * 60 + 30 = 450).

A) Per verificare se il dipendente può uscire oppure no, scrivere la funzione ***verificaUscita*** che analizza il ***registro*** e un ***ID*** e controlla se il dipendente corrispondente ha lavorato nella giornata odierna per almeno 7 ore e 30 minuti (i.e., 450 minuti). La funzione restituisce vero se questa condizione risulta verificata, falso in caso contrario.

B) Scrivere la funzione ***badge*** che prende in ingresso un ***dipendente***, la struttura ***registro***, la data odierna (rappresentata come indicato al punto 2), un intero contenente l'ora corrente (rappresentata come indicato al punto 2).
La funzione ***badge*** aggiorna ***registro*** nel seguente modo: se nella data odierna il dipendente non compare nella struttura ***registro*** (ossia l'utente sta entrando in azienda), la funzione ne aggiunge le informazioni indicando come ***data*** e ***oraIngresso*** la data e l'ora passate come parametro. Altrimenti, se il dipendente è già presente nel ***registro*** alla data odierna (ossia l'utente sta uscendo dall'azienda), la funzione ***badge*** aggiorna il campo ***oraUscita*** relativo al dipendente con il valore del parametro ora. La funzione ***badge*** quindi richiama la funzione ***verificaUscita*** descritta al punto A e, se il dipendente può uscire stampa a video *'Arrivederci!'*, altrimenti *'Errore uscita!'*

C) Si supponga che il registro sia stato precedentemente popolato con i dati relativi ai dipendenti di un'azienda. Si scriva un frammento di codice per definire una variabile ***dip1*** corrispondente al dipendente *"Giacomo Puccini"* avente ID = 1858. Si invochi inoltre la funzione ***badge*** al fine di inserire nel registro il suo ingresso il 3 Febbraio 2017 alle 8:15.

Soluzione

A)

```
function okUscita = verificaUscita(registro, ID, data)

indexDipendente = ([registro.ID] == ID) & ([registro.data] == data);
okUscita = registro(indexDipendente).oraUscita - registro(indexDipendente).oraIngresso >
450;
```

B)

```
function [registro] = badge(dipendente, registro, data, ora)

indiceUtente =([registro.ID] == dipendente.ID) & ([registro.data] == data);
if (~any(indiceUtente))
        registro(end+1).ID = dipendente.ID;
        registro(end).nome = dipendente.nome;
        registro(end).cognome = dipendente.cognome;
        registro(end).data = data;
        registro(end).oraIngresso = ora;
else
        registro(indiceUtente).oraUscita = ora;
        okUscita = verificaUscita(registro, dipendente.ID, data);
        if (okUscita)
                disp('Arrivederci!');
        else
                disp('Errore uscita!');
        end
end
```

C)

```
dip1.nome = ‘Giacomo';
dip1.cognome = 'Puccini';
dip1.ID = 1858;
registro = badge(dip1, registro, 20170203, 495);
```

Esercizio 1B (8 punti)

Una palestra ha installato un sistema che registra gli ingressi e le uscite dei propri clienti tramite l'uso di badge. All'entrata e all'uscita, ogni cliente deve registrarsi passando il badge nel lettore posto all'ingresso. Siete stati incaricati di scrivere in Matlab parte del software per la gestione degli accessi.

Il programma Matlab è costituito da una serie di funzioni che operano sui seguenti dati:

1. Informazioni relative a ciascun ***cliente***. Queste sono organizzate in una struct con i seguenti campi:
    - ***matricola*** (un numero intero che identifica univocamente ogni cliente),
    - ***nome*** (stringa)
    - ***cognome*** (stringa)
2. L'***archivio*** dei clienti. Questo è un array di struct che contiene in ogni suo elemento:
    - I campi di un cliente (***matricola, nome, cognome***).
    - La ***data*** che corrisponde a un ingresso e ad un'uscita del cliente (si suppone che avvengano nello stesso giorno). Questa data è rappresentata con un numero intero di 8 cifre decimali. Per esempio, il numero 20171223 corrisponde alla data 23 Dicembre 2017.
    - ***eserciziTotali*** ed ***eserciziSvolti***, ognuno rappresentato con un numero intero che indica rispettivamente il numero di esercizi che il cliente deve svolgere e quelli effettivamente svolti dal cliente.

A) Per stimare l'intensità dell'allenamento di un cliente si implementi la funzione ***valutaAllenamento*** che analizza ***archivio*** e riceve una ***matricola*** ed una ***data***. La funzione calcola ***intensitaAllenamento*** come la percentuale di esercizi svolti (rispetto a quelli previsti) dal cliente corrispondente a ***matricola*** nella data specificata. La funzione quindi restituisce *true* se ***intensitaAllenamento*** è superiore o uguale a 0.47 (allenamento intenso), *false* altrimenti.

B) Scrivere la funzione ***tessera*** che prende in ingresso i dati di un ***cliente***, la struttura ***archivio***, la data odierna (rappresentata come indicato al punto 2), un intero ***esercizi*** contenente il numero di esercizi da svolgere.
La funzione aggiorna la variabile ***archivio*** nel seguente modo: se nella data odierna il cliente non compare nella struttura ***archivio*** (ossia il cliente sta entrando in palestra), la funzione ***tessera*** ne aggiunge le informazioni all'***archivio*** indicando come ***data*** ed ***eserciziTotali*** il valore passati come parametro. Altrimenti, se il cliente è già presente nella struttura ***archivio*** alla data odierna (ossia il cliente sta uscendo dalla palestra), la funzione ***tessera*** aggiorna il campo ***eserciziSvolti*** relativo a quel cliente con il valore del parametro **esercizi** e calcola, richiamando la funzione ***valutaAllenamento***, se l'allenamento è stato intenso o meno. Se l'allenamento è intenso per il cliente, stampa a video "Ottimo allenamento!", altrimenti stampa a video "Allenamento poco intenso!".

C) Si supponga che **l'*archivio*** sia stato precedentemente riempito con i dati dei clienti di una palestra. Si scriva un frammento di codice per definire una variabile ***cliente1***, corrispondente al cliente "*Giuseppe Verdi*" avente matricola uguale a 1813. Si invochi

quindi la funzione ***tessera*** al fine di inserire nella variabile ***archivio*** il suo ingresso il 6 Marzo 2017 con un numero di esercizi da svolgere pari a 40.

Soluzione:

A)

```
function okAllenamento = valutaAllenamento(archivio, matricola, data)

   indexCliente =([archivio.matricola] == matricola) & ([archivio.data] == data);
   intensitaAllenamento = archivio(indexCliente).eserciziSvolti /
archivio(indexCliente).eserciziTotali;
   okAllenamento = intensitaAllenamento > 0.47;
end
```

B)

```
function [archivio] = tessera(cliente, archivio, data, esercizi)

indiceCliente = ([archivio.matricola] == cliente.matricola) & ([archivio.data] == data);
if (~any(indiceCliente))
      archivio(end+1).matricola = cliente.matricola;
      archivio(end).nome = cliente.nome;
      archivio(end).cognome = cliente.cognome;
      archivio(end).data = data;
      archivio(end).eserciziTotali = esercizi;
else
      archivio(indiceCliente).eserciziSvolti = esercizi;
      okAllenamento = valutaAllenamento(archivio, cliente.matricola, data);
      if (okAllenamento)
         disp('Ottimo allenamento!');
      else
         disp('Allenamento poco intenso!');
      end
   end
```

C)

```
cliente1.matricola = 1813;
cliente1.nome = ‘Giuseppe’;
cliente1.cognome = 'Verdi';
archivio = tessera(cliente1, archivio, '20170306', 40);
```

Esercizio 2A (6 punti)

Si scriva una funzione ricorsiva ***pallinaRic*** che stampa a schermo una sequenza di caratteri che rappresenta uno spostamento di una pallina da sinistra a destra dello schermo. L'effetto che si vuole ottenere è il seguente:

```
o------
-o-----
--o----
---o---
----o--
-----o-
------o
-------
```

La funzione ***pallinaRic*** prende in ingresso i seguenti parametri:

- ***n*** che rappresenta il numero di caratteri in ogni riga
- ***pos***, che indica la posizione attualmente occupata dalla pallina.

La visualizzazione sopra riportata è stata ottenuta dall'invocazione di ***pallinaRic****(7,1)*

A) Si definisca la funzione ***pallinaRic***. In particolare, si completino i seguenti punti:

1. Si specifichi il caso base, cioè quello che determina la terminazione della ricorsione.
2. Si specifichi il caso ricorsivo.

B) Si modifichi la funzione ***pallinaRic*** per creare la funzione ***pallinaRicConRit***, per fare in modo che la pallina ritorni alla posizione iniziale. Dall'invocazione di ***pallinaRicConRit****(7,1)* si dovrà quindi ottenere

```
o------
-o-----
--o----
---o---
----o--
-----o-
------o
-------
------o
-----o-
----o--
---o---
--o----
-o-----
o------
```

Soluzione

A)

```
function [] = pallinaRic( n, pos )

  if pos > 0 && n > 0
    if pos > n
        riga(1:n) = '-';
        disp(riga);
    else
        riga(1:n) = '-';
        riga(pos) = 'o';
        disp(riga);
        pallinaRic2(n, pos+1);
    end
  end
```

B)

```
function [] = pallinaRicConRit( n, pos )

  if pos > 0 && n > 0
    if pos > n
        riga(1:n) = '-';
        disp(riga);
    else
        riga(1:n) = '-';
        riga(pos) = 'o';
        disp(riga);
        pallinaRicConRit(n, pos+1);
        disp(riga);
    end
  end
```

Esercizio 2B (6 punti)

Si scriva una funzione ricorsiva ***disegnaVRic*** che stampa a schermo una sequenza di caratteri che rappresentano una V, come quella riportata nel seguito:

```
I . . . . . . . . . I
. I . . . . . . . I .
. . I . . . . . I . .
. . . I . . . I . . .
. . . . I . I . . . .
. . . . . I . . . . .
```

La funzione ***disegnaVRic*** prende in ingresso i seguenti parametri:

- ***n***, che rappresenta il numero di caratteri in ogni riga
- ***pos***, un intero che indica la colonna di inizio e fine della V sulla prima riga. Se, per esempio, pos è pari a 3, questo significa che la V parte dalla terza posizione sulla prima riga e termina nella terz'ultima posizione sulla stessa riga.

Ad esempio, la visualizzazione sopra riportata è stata ottenuta dall'invocazione ***disegnaVRic****(11,1)*, mentre la seguente visualizzazione è stata ottenuta dall'invocazione ***disegnaVRic****(11, 3)*

```
. . I . . . . . I . .
. . . I . . . I . . .
. . . . I . I . . . .
. . . . . I . . . . .
```

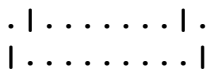
N.B Per semplicità si assuma che il primo parametro sia sempre dispari

A) Si definisca la funzione ***disegnaVRic***. In particolare, si completino i seguenti punti:

1. Si specifichi il caso base, cioè quello che determina la terminazione della ricorsione.
2. Si specifichi il caso ricorsivo.

B) A partire dalla funzione ***disegnaVRic*** si ri ricavi la funzione ***disegnaXRic*** che disegna una 'X' a schermo, facendo aprire le linee diagonali dopo che si sono incrociate. Dall'invocazione di ***disegnaXRic*** *(11,1)* si dovrà quindi ottenere:

```
I . . . . . . . . . I
. I . . . . . . . I .
. . I . . . . . I . .
. . . I . . . I . . .
. . . . I . I . . . .
. . . . . I . . . . .
. . . . I . I . . . .
. . . I . . . I . . .
. . I . . . . . I . .
```



Soluzione

A)

```
function [] = disegnaVRic(n, pos)

 if(n > 0 && pos > 0  && pos <=  ceil(n/2))
    stringa(1:n) = '.';
    stringa(pos) = '|';
    if(pos == ceil(n/2))
        disp(stringa);
    else
        stringa(end-pos+1) = '|';
        disp(stringa);
        disegnaVRic(n, pos+1);
    end
  end
```

B)

```
function [] = disegnaXRic(n, pos)

  if(n > 0 && pos > 0  && pos <=  ceil(n/2))
    stringa(1:n) = '.';
    stringa(pos) = '|';
    if(pos == ceil(n/2))
        disp(stringa);
    else
        stringa(end-pos+1) = '|';
        disp(stringa);
        disegnaXRic(n, pos+1);
        disp(stringa);
    end
  end
```

Esercizio 3A (3 punti)

A) Un sistema dispone di celle di memoria di 1 byte, di 64 Kbyte di memoria fisica indirizzabile e di 256 Kbyte di memoria virtuale indirizzabile. La memoria virtuale è organizzata in pagine di 512 byte.
Rispondere alle seguenti domande (giustificando i risultati ottenuti e riportando gli opportuni calcoli):

1. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.
2. Quante sono le pagine di memoria virtuale?

B) Si consideri un sistema con le seguenti caratteristiche:
Indirizzo virtuale di 9 bit
Indirizzo fisico di 7 bit
Dimensione pagine 16 byte

L'indirizzo virtuale 001010000 può corrispondere all'indirizzo fisico 0010100? Giustificare la risposta.

Per i calcoli si può fare riferimento alla seguente tabella delle potenze di due:

2^1	=	2	2^{11}	=	2 048	2^{21}	=	2 097 152	2^{31}	=	2 147 483 648
2^2	=	4	**2^{12}**	**=**	**4 096**	2^{22}	=	4 194 304	**2^{32}**	**=**	**4 294 967 296**
2^3	=	8	2^{13}	=	8 192	2^{23}	=	8 388 608	2^{33}	=	8 589 934 592
2^4	**=**	16	2^{14}	=	16 384	**2^{24}**	**=**	**16 777 216**	2^{34}	=	17 179 869 184
2^5	=	32	2^{15}	=	32 768	2^{25}	=	33 554 432	2^{35}	=	34 359 738 368
2^6	=	64	**2^{16}**	**=**	**65 536**	2^{26}	=	67 108 864	**2^{36}**	**=**	**68 719 476 736**
2^7	=	128	2^{17}	=	131 072	2^{27}	=	134 217 728	2^{37}	=	137 438 953 472
2^8	**=**	256	2^{18}	=	262 144	**2^{28}**	**=**	**268 435 456**	2^{38}	=	274 877 906 944
2^9	=	512	2^{19}	=	524 288	2^{29}	=	536 870 912	2^{39}	=	549 755 813 888
2^{10}	=	1 024	**2^{20}**	**=**	**1 048 576**	2^{30}	=	1 073 741 824	**2^{40}**	**=**	**1 099 511 627 776**

Soluzione

A1)
Struttura indirizzo virtuale
256 KB = 2^18 B -> Indirizzo di 18 bit
Bit offset = log2(512) = 9 -> 9 bit di offset
Il numero di bit di offset è uguale sia per l'indirizzo fisico che per quello virtuale
Numero pagine = 256 KB / 512 byte = 2^18 / 2^9 = 2^9
Bit numero pagine = log2(#Pagine) = log2(2^9) = 9 (oppure dato che l'indirizzo è di 18 bit e l'offset di 9, basta fare 18 – 9 = 9)
Struttura indirizzo fisico: 64 KB = 2^16 byte -> Indirizzo di 16 bit
Bit offset = 9
Bit numero pagine = 16 – 9 = 7

A2)
Il numero di pagine virtuali e' 2^9 = 512

B)
Bit offset = log2(16) = 4
L'offset in entrambi gli indirizzi virtuali e fisici e' di 4 bit (i 4 bit più a destra).
I due indirizzi possono corrispondere se e solo se gli ultimi 4 bit sono tra loro uguali:
Ultimi 4 bit indirizzo virtuale: 0000
Ultimi 4 bit indirizzo fisico: 0100
I valori dei bit di offset nei due indirizzi differiscono tra loro, quindi i due indirizzi non possono corrispondere.

Esercizio 3B (3 punti)

A) Un sistema dispone di celle di memoria di 1 byte, di 128 Kbyte di memoria fisica indirizzabile e di 256 Kbyte di memoria virtuale indirizzabile. La memoria virtuale è organizzata in pagine di 1 Kbyte.
Rispondere alle seguenti domande (giustificando i risultati ottenuti e riportando gli opportuni calcoli):

1. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.
2. Quante sono le pagine di memoria virtuale?

B) Si consideri un sistema con le seguenti caratteristiche:
Indirizzo virtuale di 10 bit
Indirizzo fisico di 6 bit
Dimensione pagine 32 byte

L'indirizzo virtuale 1001010100 può corrispondere all'indirizzo fisico 010100? Giustificare la risposta.

Per i calcoli si può fare riferimento alla seguente tabella delle potenze di due:

2^1	=	2	2^{11}	=	2 048	2^{21}	=	2 097 152	2^{31}	=	2 147 483 648
2^2	=	4	**2^{12}**	**=**	**4 096**	2^{22}	=	4 194 304	**2^{32}**	**=**	**4 294 967 296**
2^3	=	8	2^{13}	=	8 192	2^{23}	=	8 388 608	2^{33}	=	8 589 934 592
2^4	**=**	16	2^{14}	=	16 384	**2^{24}**	**=**	**16 777 216**	2^{34}	=	17 179 869 184
2^5	=	32	2^{15}	=	32 768	2^{25}	=	33 554 432	2^{35}	=	34 359 738 368
2^6	=	64	**2^{16}**	**=**	**65 536**	2^{26}	=	67 108 864	**2^{36}**	**=**	**68 719 476 736**
2^7	=	128	2^{17}	=	131 072	2^{27}	=	134 217 728	2^{37}	=	137 438 953 472
2^8	**=**	256	2^{18}	=	262 144	**2^{28}**	**=**	**268 435 456**	2^{38}	=	274 877 906 944
2^9	=	512	2^{19}	=	524 288	2^{29}	=	536 870 912	2^{39}	=	549 755 813 888
2^{10}	=	1 024	**2^{20}**	**=**	**1 048 576**	2^{30}	=	1 073 741 824	**2^{40}**	**=**	**1 099 511 627 776**

Soluzione

A1)
Struttura indirizzo virtuale:
256 KB = 2^18 B -> Indirizzo di 18 bit
Bit offset = log2(1024) = 10 -> 10 bit di offset
Il numero di bit di offset è uguale sia per l'indirizzo fisico che per quello virtuale
Numero pagine = 256 KB / 1 KB = 2^18 / 2^10 = 2^8
Bit numero pagine = log2(#Pagine) = log2(2^8) = 8 (oppure dato che l'indirizzo è di 18 bit e l'offset di 10, basta fare 18 – 10 = 8)
Struttura indirizzo fisico: 128 KB = 2^17 byte -> Indirizzo di 17 bit
Bit offset = 10
Bit numero pagine = 17 – 10 = 7

A2)
Il numero di pagine virtuali e' 2^8 = 256

B)
Bit offset = log2(32) = 5
L'offset in entrambi gli indirizzi virtuali e fisici e' di 5 bit (i 5 bit più a destra).
I due indirizzi possono corrispondere se e solo se gli ultimi 5 bit sono tra loro uguali:
Ultimi 5 bit indirizzo virtuale: 10100
Ultimi 5 bit indirizzo fisico: 10100
I valori dei bit di offset nei due indirizzi sono uguali, quindi i due indirizzi possono corrispondere.