



Politecnico di Milano

Facoltà di Ingegneria Civile, Ambientale e Territoriale

Informatica [097256] [091461] [091456]

Prof. G. Boracchi

Allievi Ingegneria Civile e Ambientale

Prima prova in itinere (14 Novembre 2017)

<i>Cognome e nome</i>	
<i>Matricola</i>	
<i>Firma</i>	

Domanda	1	2	3	4	5	TOT
Punteggio max	10	10	6	3	3	32
Punteggio						

La **durata** della **prova** è di **2 ore e mezza**. Non è consentito consultare libri o appunti, non è consentito l'uso di calcolatrici.

Scrivere solo sui fogli distribuiti utilizzando il retro delle pagine in caso di necessità e cancellando le parti di brutta con un tratto di penna. Non separare questi fogli. Non è necessario trascrivere in penna, verranno valutate anche le parti a matita.

Per tutti gli esercizi non è sufficiente fornire il risultato, ma **è necessario mostrare il procedimento seguito. Non dilungarsi però in spiegazioni prolisse.**

Gli allievi sono invitati a privilegiare **chiarezza, proprietà di linguaggio e sinteticità** nelle risposte agli esercizi, con l'obiettivo di **dimostrare la loro conoscenza degli argomenti.**

Domanda 1 (10 punti)

Si definiscano le seguenti variabili

```
tubo = struct('diametro', [], 'materiale', []);  
% dove materiale può essere 'plastica', 'ferro', oppure 'alluminio'  
  
valvola = struct('diametroIn', [], 'diametroOut', [], ...  
'materiale', []);  
% dove materiale può essere 'plastica', 'ferro', oppure 'alluminio'
```

Il vostro compito è popolare l'array `valvoleScelte` con gli elementi adatti a connettere una tubatura da 100 tubi. I tubi da utilizzare sono già stati e ordinati in un vettore di strutture chiamato `tubiScelti`.

In particolare, `tubiScelti(1), ... , tubiScelti(100)` sono strutture come `tubo` già popolate in entrambi i campi. L'ordine delle strutture nell'array corrisponde all'ordine con cui sono posizionati i tubi nella tubatura. Quindi la struttura `tubiScelti(ii)` descrive l'*ii*-simo elemento della tubatura.

Si risolva il problema sviluppando le seguenti funzioni:

1) funzione `controllaTubatura` che restituisce `true` se l'array `tubiScelti` non contiene nemmeno un tubo in plastica, `false` altrimenti.

2) funzione `acquistaValvole` che restituisce un array `valvoleScelte` di 99 elementi così definiti:

- la `valvoleScelte(ii)` viene utilizzata per collegare `tubiScelti(ii)` e `tubiScelti(ii+1)`
- il campo `diametroIn` di `valvoleScelte(ii)` è pari al valore `diametro` di `tubiScelti(ii)`, `diametroOut` è pari a `diametro` di `tubiScelti(ii + 1)`.
- il campo `materiale` di `valvoleScelte(ii)` è definito come il materiale più robusto tra quelli di `tubiScelti(ii)` e `tubiScelti(ii + 1)`. Il materiale dal più robusto al meno robusto sono: ferro, alluminio, plastica.

3) funzione `selezionaValvole` che rimuove dall'array `valvoleScelte` tutte le valvole che collegano tubi di uguale spessore.

Per ogni funzione scrivere

1. il prototipo (precisando tutti i parametri formali in ingresso ed in uscita),
2. il corpo
3. un esempio di invocazione

Hint

```
>> help strcmp  
strcmp Compare strings or character vectors  
TF = strcmp(S1, S2) compares two strings S1 and S2 and returns  
logical 1 (true) if they are identical, and returns logical 0  
(false) otherwise.
```

Soluzione

```
function res = controllaTubatura(tubiScelti)
res = true;
for ii = 1 : 100
    if strcmp(tubiScelti(ii).materiale, 'plastica')
        res = false;
    end
end
end

function valvole = acquistaValvole(tubiScelti)
% non necessaria questa definizione
valvole = struct('diametroIn', [], 'diametroOut', [], 'materiale', []);
for ii = 1 : 99
    valvole(ii).diametroIn = tubiscelti(ii).diametro;
    valvole(ii).diametroOut = tubiscelti(ii + 1).diametro;

    if strcmp(tubiScelti(ii).materiale, 'ferro') || strcmp(tubiScelti(ii + 1).materiale,
'ferro')
        valvole(ii).materiale = 'ferro';
    elseif strcmp(tubiScelti(ii).materiale, 'alluminio') || strcmp(tubiScelti(ii +
1).materiale, 'alluminio')
        valvole(ii).materiale = 'alluminio';
    else
        valvole(ii).materiale = 'plastica';
    end
end
end

function valvole = selezionaValvole(valvoleScelte)
indxToRemove = []; % attenzione che bisogna rimuoverle alla fine, non durante, altrimenti
non
    % si può usare il for
for ii = 1 : length(valvoleScelte)
    if(valvoleScelte(ii).diametroIn == valvoleScelte(ii).diametroOut)
        indxToRemove = [indxToRemove, ii];
    end
end
valvole = valvoleScelte;
valvole(indxToRemove) = [];

end
```

Domanda 2 (10 punti)

Si sviluppino le seguenti funzioni (attenzione ai parametri formali in ingresso e uscita):

1. funzione `inserimentoControllato` che chiede all'utente di inserire un vettore V di 100 elementi eseguendo il seguente controllo:
 - a. Il primo valore del vettore deve essere maggiore di zero
 - b. ogni valore successivo deve essere maggiore del valore alla posizione precedente.
2. funzione `creaMatrice` che crea una matrice M di dimensione 100×100 la cui prima colonna è composta dal vettore V , mentre le colonne successive hanno valori pari al valore della colonna precedente incrementata o decrementata di un valore casuale uniformemente distribuito tra -5 a 5.
3. funzione `mediaRighe` che riceve in ingresso una matrice X (di dimensione 10×10) e restituisce un vettore di dimensione 1×10 il cui ii -esimo valore corrisponde al valore medio degli elementi della riga ii -esima della matrice X .
4. Si scriva quindi uno script in cui si
 - a. invoca la funzione `inserimentoControllato` per creare il vettore V
 - b. invoca la funzione `creaMatrice` sul vettore inserito
 - c. invoca la funzione `mediaRighe` sulla sottomatrice (di dimensione 10×10) in alto a sinistra di M
 - d. invoca la funzione `mediaRighe` sulla sottomatrice (di dimensione 10×10) in basso a destra di M
5. Nello script, si utilizzi la funzione `plot` per rappresentare il vettore V utilizzando
 - a. un simbolo 'o' blu in corrispondenza dei valori alle posizioni dispari di V
 - b. un simbolo '+' rosso in corrispondenza dei valori pari di V
 - c. un simbolo '.' verde in corrispondenza dei valori in cui V è minore della seconda colonna di M

Hint

```
>> help randi
```

```
...
```

```
R = randi([IMIN,IMAX],M,N) returns an M-by-N matrix containing  
integer values drawn from the discrete uniform distribution on  
IMIN:IMAX.
```

```
...
```

Soluzione

```
function V = inserimentoControllato()

V(1) = -1;
while(V(1) <= 0)
    temp = input(['inserire un numero positivo:']);
    V = temp(1);
end

for ii = 2 : 100
    V(ii) = -1;
    while(V(ii) <= V(ii - 1))
        temp = input(['inserire numero (pos ', num2str(ii), '):']);
        V(ii) = temp(1);
    end
end

function M = creaMatrice(V)

M(:, 1) = V';

for ii = 2 : 100
    M(:, ii) = M(:, ii - 1) + randi([-5, 5], 100, 1);
end

end

function R = mediaRighe(X)

R = mean(X, 2);

% script
close all
clear
clc

V = inserimentoControllato();
M = creaMatrice(V);
R1 = mediaRighe(M(1 : 10, 1: 10));
R2 = mediaRighe(M(end - 10 : end, end - 10: end));

figure(1),

xx = [1 : 2 : 100];
plot(xx, V(xx), 'bo')
hold on
```

```
indx = find(mod(V, 2) == 0);  
plot(indx, V(indx), 'r+')  
colonna =M(:, 2);  
indx = find(V < colonna')  
plot(indx, V(indx), 'g.')
```

Domanda 3 Matlab (6 punti).

Si consideri la seguente funzione Matlab

```
function vet = chefa(vet)

if isempty(vet) || length(vet) == 1
    return
end

if all(vet(1) >= vet(2 : end)) % punto 1) del codice
    return
end

vet(vet == vet(1)) = [];
vet = chefa(vet);
```

si fornisca l'output delle seguenti invocazioni

```
>> chefa([19 18 12 19 12 3])
```

```
>> chefa([7 18 13 19 12 3])
```

```
>> chefa([ 2 3 18 12 19 12 3])
```

si descriva **brevemente** cosa restituisce un'invocazione di chefa.m

si dica **brevemente** cosa succederebbe se modificassimo la condizione alla riga %1) con
`all(vet(1) > vet(2 : end))`

Cosa restituirebbe la seguente invocazione?

```
>> chefa([19 18 12 19 12 3])
```

Soluzione

```
>> chefa([ 19 18 12 19 12 3])
```

```
ans =
```

```
19 18 12 19 12 3
```

```
>> chefa([7 18 13 19 12 3])
```

```
ans =
```

```
19 12 3
```

```
>> chefa([ 2 3 18 12 19 12 3])
```

```
ans =
```

```
19
```

La funzione rimuove restituisce un sottovettore dell'input avente il primo elemento maggiore dei successivi. La funzione ricerca i sottovettori rimuovendo ad ogni invocazione il primo elemento. Inoltre, qualora vi fossero più occorrenze di un elemento diverso dal massimo del vettore, tutte queste vengono rimosse.

Se modificassi la condizione 1) come indicato, avremmo che anche il massimo del vettore viene rimosso se questo non è unico.

```
chefa([ 19 18 12 19 12 3])
```

```
ans =
```

```
18 12 12 3
```

Domanda 4 Matlab (3 punti).

Si consideri il seguente script e si riporti qualsiasi carattere stampato a schermo.

```
vet1 = [7 : -1 : 1];
disp(['vet1 = ', num2str(vet1), ''])

vet2 = [-6 : 2 : 6];
disp(['vet2 = ', num2str(vet2), ''])

a = vet1 <= vet2.^2;
disp(['a = ', num2str(a), ''])

b = vet1 > vet2;
disp(['b = ', num2str(b), ''])

out = [];
for k = find(a & b)
    fprintf('\nk = %2.2f', k);
    out = [out, vet1(k) + vet2(k)];
end

disp(['out = ', num2str(out), ''])
```

Soluzione

```
vet1 = [7  6  5  4  3  2  1]
vet2 = [-6 -4 -2  0  2  4  6]
a = [1  1  0  0  1  1  1]
b = [1  1  1  1  1  0  0]
k = 1.00
k = 2.00
k = 5.00
out = [1  2  5]
```

Domanda 5 Matlab (3 punti).

Si dica se i seguenti codici sono equivalenti

```
%% Codice 1)
a = -1;
b = a;
c = 3;

while ~(a > b) && (b < 3) || c ~= 2)
    a = input(['inserire a']);
    a = a(1);
    b = input(['inserire b']);
    b = b(1);
    c = input(['inserire c']);
    c = c(1);
end

%% Codice 2)
a = -1;
b = a;
c = 3;

while ~(c == 2 && ~((a <= b) && (b < 3)))
    a = input(['inserire a']);
    a = a(1);
    b = input(['inserire b']);
    b = b(1);
    c = input(['inserire c']);
    c = c(1);
end
```

Hint: trattare la condizione del while come un'espressione booleana composta

Soluzione

I due codici coincidono se le condizioni di permanenza nei cicli while coincidono (le restanti istruzioni sono equivalenti).

Definiamo le seguenti espressioni logiche

$$X = (a > b)$$

$$Y = (b < 3)$$

$$Z = (c \neq 3)$$

La condizione di permanenza nel primo while è $\sim X \ \&\& \ Y \ || \ Z$

La condizione di permanenza nel secondo while è $\sim (\sim Z \ \&\& \ \sim (\sim X \ \&\& \ Y))$

Si può dimostrare che le due coincidono per le Leggi di De Morgan o costruendo una tabella di verità che dimostra che le due hanno pari output quando gli input coincidono

Applicando de Morgan si ottiene

$$\sim (\sim Z \ \&\& \ \sim (\sim X \ \&\& \ Y)) =$$

$$(Z \ || \ (\sim X \ \&\& \ Y)) =$$

$$((\sim X \ \&\& \ Y) \ || \ Z) =$$

$$\sim X \ \&\& \ Y \ || \ Z$$

