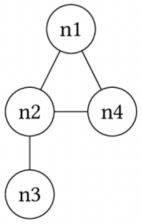
11 Esercizi di Riepilogo

11.1 Esercizi "tipo" esame

Esercizio 11.1

(TdE Gennaio 2019)

Una `cricca` è un insieme di amici che si conoscono tutti fra di loro. Se si modella la relazione di amicizia con un insieme di nodi (amici) e archi fra essi (ognuno che indica che le due persone (nodi) collegate dall'arco si conoscono), una cricca è visibile poiché i relativi nodi sono tutti connessi fra di loro. Ad esempio, supponiamo di avere quattro persone n1, n2, n3, n4 e che sussistano le seguenti relazioni di amicizia:



Diciamo allora che il sottoinsieme {n1, n2, n4} di persone forma una cricca di cui n3 non fa parte perché è amico solo di n2.

1) In linguaggio Matlab, si scriva una funzione `iscricca(v, m)` che riconosca gli insiemi di persone che formano una cricca. Essa riceve in input due parametri e restituisce un solo valore:

- Il vettore `v` in input, con un numero di elementi pari al numero di persone considerate, utilizzato per specificare l'insieme di persone di cui ci si chiede se formano una cricca. Esso ha un 1 in posizione ii se stiamo indagando una cricca che contiene ii altrimenti ha uno 0. Ad esempio, il vettore v = $[1, 1, 0, 1]$ rappresenta il sottoinsieme {n1, n2, n4} ed esclude n3.
- Una matrice quadrata `m` in input, con numero di righe e colonne pari al numero di persone dell'insieme considerato. Per ogni riga ii e colonna jj vi è un 1 nella

matrice se le persone rappresentate da ii e jj si conoscono, altrimenti vi è uno 0; per definizione, ogni persona è amica di se stessa. Nel nostro esempio, ipotizzando di nuovo l'ovvia corrispondenza tra gli indici della matrice e il nome degli amici (n1 corrisponde all'indice 1 e così via), avremo la seguente matrice:

$$m = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

- Il valore restituito `r` dalla funzione vale 1 se `v` indica un insieme di persone che effettivamente rappresenta una cricca, 0 altrimenti. Nel nostro esempio con v = $[1, 1, 0, 1]$, `iscricca(v, m)` restituisce 1, poiché {n1, n2, n4} formano una cricca.

Per implementare la funzione si suggerisce il seguente algoritmo (Moody & Hollis):

1. Per ogni riga ii di m, se v(ii) = 0, sostituirla con la riga i-sima di ones(n) - eye(n) (dove n è il numero totale di persone);
2. Se poi l'and logico di tutte le righe di m produce il vettore v originario, allora il vettore v in input rappresenta una cricca, altrimenti no.

2) Si scriva una porzione di codice Matlab che legga da un file data.mat la matrice m, richieda all'utente il valore di un vettore v e controlli, segnalando a video l'esito, se il vettore v rappresenta una cricca o meno.

Esercizio 11.2

(TdE Luglio 2020)

1) Scrivere una funzione di nome `corona` che riceve un parametro N (intero pari positivo) e restituisce una matrice `A` di dimensione N×N in cui:

- gli elementi della riga 1, riga N, colonna 1 e colonna N sono inizializzati a 1;
- gli elementi rimanenti della riga 2, riga N - 1, colonna 2 e colonna N - 1 sono inizializzati a 2;
- gli elementi rimanenti della riga 3, riga N - 2, colonna 3 e colonna N - 2 sono inizializzati a 3;
- così via fino a completare tutta la matrice.

Ad esempio, per N = 8:

```
1 corona(8)
```

```
2
3  ans =
4
5       1       1       1       1       1       1       1       1
6       1       2       2       2       2       2       2       1
7       1       2       3       3       3       3       2       1
8       1       2       3       4       4       3       2       1
9       1       2       3       4       4       3       2       1
10      1       2       3       3       3       3       2       1
11      1       2       2       2       2       2       2       1
12      1       1       1       1       1       1       1       1
```

Nota: una soluzione ottimizzata risolve il problema con un solo ciclo for. Non saranno valutate positivamente le soluzioni che usano più di 2 cicli for annidati.

2) Scrivere uno script che dopo aver generato un numero randomico tra 5 e 20, utilizzi tale numero per creare una corona M di tali dimensioni. Si stampi a video un'immagine contenente la corona M e si aggiunga un titolo alla figura. Infine si calcoli la somma di tutti gli elementi in M e si salvi il risultato in un file somma.mat.

Esercizio 11.3

(TdE Febbraio 2018)

Un'agenzia di trading online vuole memorizzare l'andamento del valore dei titoli che controlla. La memorizzazione viene effettuata in **500** istanti temporali equidistanti. I dati vengono salvati nel file `MATLAB` **log.mat** che contiene:

- la matrice **titoli** le cui righe rappresentano i diversi titoli controllati e le cui colonne rappresentano i vari istanti in cui sono stati memorizzati i valori di tali titoli (quindi ogni cella della matrice contiene il valore di un titolo in un dato istante);
- il vettore colonna **andamento** con lo stesso numero di righe della matrice **titoli** che contiene un valore numerico per ogni titolo, indicativo del suo andamento complessivo crescente o decrescente.

1) Scrivere in linguaggio `MATLAB` una funzione `splittaMatrice` che:

- riceva in input una matrice **titoliTot** (con la stessa struttura di **titoli**), un vettore **andamentoTot** (con stessa struttura di vettore **andamento**) e uno scalare **soglia**;
- fornisca in output due matrici **titoliOver** e **titoliUnder** (ognuna con la stessa struttura di **titoliTot**). **titoliOver** include solo le righe di **titoliTot** corrispondenti agli elementi di **andamentoTot** con valore maggiore o uguale di **soglia**. **titoliUnder**, invece, include le righe di **titoliTot** corrispondenti agli elementi di **andamentoTot**

con valore minori di **soglia**.

2) Scrivere in linguaggio MATLAB uno script che:

A) legga dal file **log.mat** i due dati memorizzati: **titoli** e **andamento**;

B) richiami la funzione `splittaMatrice` per separare **titoli** nelle due matrici **titoliOver** e **titoliUnder**, per un valore di **soglia** pari a 0;

C) crei un vettore **x** che contenga i 500 istanti di memorizzazione;

D) disegni su due grafici separati (che includano il titolo del grafico e il nome dei due assi) l'andamento dei titoli in **titoliOver** e **titoliUnder**, in funzione di **x**.

11.2 Esercizi vari

Esercizio 11.4

Scrivere uno script in MATLAB che esegua le seguenti operazioni:

1. Legge da tastiera due valori positivi minori di 100 che vengono memorizzati in due variabili `m` e `n`;
2. Genera una matrice `A` di dimensioni $m \times n$ di numeri casuali tra 5 e 10;
3. Costruisce una matrice `P` contenente solo le righe con indice pari della matrice `A` e una matrice `D` contenente solo le righe con indice dispari della matrice `A`;
4. se le matrici `P` e `D` hanno le stesse dimensioni, calcola anche la matrice somma.

Esercizio 11.5

Scrivere in MATLAB una funzione per analizzare i codici IBAN dei conti correnti. Un codice IBAN è una sequenza di 27 caratteri alfanumerici così composta:

- 2 caratteri maiuscoli (sigla della nazione)
- 2 cifre (CIN Europeo)
- 1 carattere maiuscolo (CIN italiano)
- 5 cifre (ABI)
- 5 cifre (CAB)
- 12 cifre (numero di conto corrente)

Si scrivano prima le seguenti tre funzioni:

- `remove_spaces`, che prende in ingresso `str_in` e restituisce `str_out` contenente tutti i caratteri di `str_in` tranne gli spazi.
- `all_upper`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene soltanto caratteri maiuscoli, 0 altrimenti.
- `all_digit`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene solo caratteri corrispondenti a cifre, 0 altrimenti.

Si usino poi tali funzioni per scrivere la funzione `check_iban` che richiede all'utente l'inserimento di un codice IBAN e restituisce 1 solo se, una volta tolti gli spazi dalla stringa IBAN, essa rispetta lo schema previsto.

Esercizio 11.6

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 5×5 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`:

0	1	1	1	1
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	1	1	1	0

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.

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.

Per esempio, nel campo di gioco presentato sopra avremo:

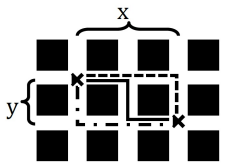
```
La riga 1 contiene una nave lunga 4
La riga 3 contiene una nave lunga 1
La riga 4 contiene una nave lunga 2
La riga 5 contiene una nave lunga 4
Sono presenti 4 navi
Lunghezza nave più corta trovata: 1
Lunghezza nave più lunga trovata: 4
Numero di navi lunghe 1: 1
Numero di navi lunghe 2: 1
Numero di navi lunghe 3: 0
Numero di navi lunghe 4: 2
Numero di navi lunghe 5: 0
```

Esercizio 11.7

Le strade della città di Grigliopoli sono organizzate come una griglia (alcune strade attraversano la città da est a ovest e altre da nord a sud). Dati due incroci che distano X isolati lungo l'asse est-ovest della città e Y isolati lungo l'asse nord-sud, si vuole calcolare il numero di percorsi a distanza minima che collegano i due incroci.

Implementare una funzione ricorsiva `calcola` in `MATLAB` che ricevuti X e Y in ingresso restituisce il numero di percorsi trovati in uscita.

Suggerimento: Quando $X = 0$ o $Y = 0$, c'è soltanto un cammino a distanza minima che collega i due incroci. Altrimenti, esiste più di un cammino minimo dal momento che è possibile sia avvicinarsi alla destinazione lungo l'asse est-ovest (riducendo quindi la distanza X) oppure avvicinarsi lungo l'asse nord-sud (riducendo la distanza Y).



Esercizio 11.8

Scrivere un programma per simulare il gioco della roulette.

La roulette possiede 38 numeri (da 1 a 36, lo zero e il doppiozero). 0 (zero) e 00 (doppiozero) non sono ne pari ne dispari (vince il banco). Inizialmente, banco e giocatori possiedono 5000 euro ciascuno.

Implementare la simulazione di una serie di giocate di due giocatori Pippo e Pluto, che giocano seguendo le seguenti strategie:

- ad ogni giocata il giocatore Pippo punta 5 euro su pari o dispari con stessa probabilità. Se vince ottiene 2 volte la posta, se perde il banco incassa il valore giocato;
- ad ogni giocata il giocatore Pluto punta 1 euro sul 15 (se esce 15 vince 36 volte la posta).

Il gioco termina quando o il banco viene sbancato (arriva a 0 euro) o entrambi i giocatori non hanno più soldi per fare la propria puntata.

Si tenga traccia delle somme a disposizione di ogni giocatore e del banco ad ogni giocata dall'inizio del gioco fino alla sua fine. Grazie a queste informazioni, disegnare l'evoluzione della disponibilità monetaria dei due giocatori e del banco. Si disegnino i valori con delle linee di spessore 2, in rosso per Pippo, in blu per Pluto e in nero per il banco. Si disegni la legenda, il titolo e si forniscano le etichette per gli assi x e y.

11.3 Esercizi strutture

Esercizio 11.9

(TdE Gennaio 2010) Un supermercato ha memorizzato il proprio archivio di scontrini nel file `scontrini.mat`, che contiene l'array di strutture `scontrini` i cui elementi hanno i seguenti campi:

- `IDcliente`: id numerico del cliente
- `totale`: totale della spesa in Euro
- `punti`: punti premio extra associati alle promozioni

Per ogni spesa, viene assegnato al cliente un quantitativo di punti premio pari alla somma dei punti più un ulteriore punto premio per ogni 10 euro spesi.

Scrivere uno script in `MATLAB` che legge il file `scontrini.mat` e costruisce un opportuno array struttura `saldo`, contenente per ciascun cliente, l'ID del cliente e il totale dei suoi punti premio. Infine, si costruisca un array `statistiche` di tre elementi rispettivamente pari al numero di clienti che possiede meno di 1000 punti premio, il numero di clienti con più di 1000 punti ma meno di 5000 punti e infine il numero di clienti con più di 5000 punti.

Nota: Si faccia attenzione al fatto che un cliente può comparire in più di uno scontrino.

Esercizio 11.10

Scrivere un programma che chieda all'utente di inserire una serie di dati contenenti ognuno i seguenti attributi:

- città (stringa)
- giorno (intero positivo)
- mese (intero positivo)
- anno (intero positivo)
- tipo di misurazione (char)
- valore (reale)

Ad esempio, l'utente potrà inserire:

```
1 Milano
2 04
3 12
4 2012
5 10.5
6 N
```

Dopo aver acquisito una certa quantità di dati, il programma dovrà chiedere all'utente il nome di una città e un tipo di misurazione. A questo punto il programma cercherà nell'archivio tutti i record riguardanti la città e il tipo di misurazione richiesti. Stamperà poi a video i dati selezionati ed il relativo valore minimo, massimo e medio dei valori.

Esercizio 11.11

Scrivere un programma per la gestione di un magazzino dove ogni prodotto nel magazzino è univocamente identificato da un codice a barre (un numero intero).

Il software di gestione associa ad ogni prodotto un carattere che indica la tipologia del prodotto e due numeri, il primo che indica il numero di pezzi in stock il secondo che indica il numero di pezzi ordinati.

Si ipotizzi che codice a barre, tipo, stock, ed ordine siano 4 vettori, già popolati, contenenti tutte le informazioni necessarie per la gestione del magazzino (l'i-esimo elemento di stock e di ordine rappresentano le quantità relative al prodotto a cui è associato l'i-esimo elemento del vettore dei codici a barre).

Ad esempio un magazzino popolato sarà:

```
barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
stock = [0 ; 300 ; 5 ; 6 ; 0 ];
ordine = [23 ; 100 ; 2 ; 100 ; 0 ];
```

Si scriva:

- la funzione `ricerca` che prende in ingresso un codice a barre ed i vettori rappresentanti il magazzino e restituisce un messaggio contenente il tipo di prodotto, il numero di pezzi in stock ed in ordine;
- un esempio di chiamata alla funzione `ricerca`;
- la funzione `ricercaMancanti` che, dato un parametro P ed il magazzino, restituisce al programma chiamante un vettore contenente i codici a barre dei prodotti:
  - se $P = 0$, non presenti in stock ma in ordine;
  - se $P = 1$, non presenti in stock che non sono nemmeno in ordine;
  - se $P = 2$, per cui ci sono più pezzi in ordine che attualmente in stock;
- un esempio di chiamata alla funzione `ricercaMancanti`;
- la funzione `aggiungiProdotto`, che permette di aggiungere al magazzino un nuovo prodotto (barcode, stock ed ordine);
- esempio di chiamata alla funzione `aggiungiProdotto`.

Soluzioni

Soluzione dell'esercizio 11.1

```
%% soluzione "intuitiva"
function r = iscricca(v, m)
    v = logical(v);
    r = all(m(v, v));
end

%% soluzione "meccanica"
function r = iscricca(v, m)
    n = length(v);
    M = ones(n) - eye(n);

    log_and = ones(1, n);
    for ii = 1:n
        if v(ii) == 0
            log_and = log_and & M(ii, :);
        else
            log_and = log_and & m(ii, :);
        end
    end

    r = all(log_and == v);
end
```

```
clear
clc

load data.mat m

[r, c] = size(m);

v = input(['Inserire vettore di dimensione ' num2str(c) ': ']);

esito = iscricca(v, m);

if esito
    disp(['Il vettore ' num2str(v) ' rappresenta una cricca']);
else
    disp(['Il vettore ' num2str(v) ' non rappresenta una cricca
       ']);
```

```
16 end
```

Soluzione dell'esercizio 11.2

```
1  function A = corona(N)
2      A = zeros(N,N);
3      for ii = 1:N/2
4          A(ii:N+1-ii, ii:N+1-ii) = ii;
5      end
6  end
```

```
1  clear
2  clc
3  close all
4
5  n = randi([5, 20]);
6
7  M = corona(n);
8
9  figure();
10 imagesc(M);
11 title('corona');
12
13 somma = sum(M(:));
14
15 save somma.mat somma;
```

Soluzione dell'esercizio 11.3

```
1  function [titoliOver, titoliUnder] ...
2                      = splittaMatrice(titoliTot, andamentoTot,
                          soglia)
3
4      titoliOver = titoliTot(andamentoTot >= soglia, :);
5      titoliUnder = titoliTot(andamentoTot < soglia, :);
6
7  end
```

```
1  close all
2  clear
3  clc
4
```

```
% A)
load log.mat titoli andamento

% B)
[over, under] = splittaMatrice(titoli, andamento, 0);

% C)
x = 1:500;

% D)
figure();
plot(x, over, 'g');
xlabel('tempo');
ylabel('valore');
title('stonks');

figure();
plot(x, under, 'r');
xlabel('tempo');
ylabel('valore');
title('not stonks');
```

Soluzione dell'esercizio 11.4

```
clear
clc
close all

m = 200;
while m > 100 || m < 1
    m = input('inserire il numero di righe: ');
end

n = 200;
while n > 100 || n < 1
    n = input('inserire il numero di colonne: ');
end

A = round(rand(m, n) * 5) + 5

P = A(2:2:m,:)
D = A(1:2:m,:)

```

```
20 if mod(m, 2) == 0
21     B = P + D
22 end
```

Soluzione dell'esercizio 11.5

```
1 clear
2 clc
3 close all
4 %IBAN di esempio: IT 02 L 12345 12345 123456789012
5 if check_iban()
6     disp('IBAN valido')
7 else
8     disp('IBAN non valido')
9 end
```

```
1 function str_out = remove_spaces(str_in)
2     str_out = str_in(str_in ~= ' ');
3 end
```

```
1 function str_out = all_upper(str_in)
2     str_out = all(str_in >= 'A' & str_in <= 'Z');
3 end
```

```
1 function r = all_digit(str_in)
2     r = all(str_in >= '0' & str_in <= '9');
3 end
```

```
1  function is_valid = check_iban()
2      % Inserimento IBAN
3      iban = input('Inserire IBAN: ', 's');
4
5      % Rimuovo spazi
6      iban = remove_spaces(iban);
7
8      % Controllo validita'
9      is_valid = all_upper(iban([1, 2, 5])) & all_digit(iban([3,
           4, 6:end])) & ...
10         length(iban) == 27;
11 end
```

Soluzione dell'esercizio 11.6

```
clear
clc
close all


%CampoGioco = [0 1 1 1 1; 0 0 0 0 0; 0 0 1 0 0; 0 1 1 0 0; 1 1
   1 1 0];
CampoGioco = input('Inserire il campo di gioco: ');

dim_gioco = size(CampoGioco,1);
nave = sum(CampoGioco,2);

for ii = 1:dim_gioco
    if nave(ii) > 0
        disp(['La riga ' num2str(ii) ' contiene una nave lunga
           ' num2str(nave(ii))]);
    end
end

nave = nave(nave> 0);

disp(['Sono presenti ' num2str(length(nave)) ' navi']);

disp(['Lunghezza nave piu'' corta trovata: ' num2str(min(nave))
   ]);
disp(['Lunghezza nave piu'' lunga trovata: ' num2str(max(nave))
   ]);

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

Soluzione dell'esercizio 11.7

```
clear
clc
close all

X = 2;
Y = 3;

```

```
calcola(X,Y)
```

```
function n_strade = calcola(X,Y)

if X == 0 || Y == 0
    n_strade = 1;
else
    n_strade = calcola(X-1,Y) + calcola(X,Y-1);
end
```

Soluzione dell'esercizio 11.8

```
clear
close all
clc

cifra_iniziale = 50;

banco = cifra_iniziale;
storicoBanco = cifra_iniziale;

giocatore.nome = 'Pippo';
giocatore.budget = cifra_iniziale;
giocatore.posta = 5;
giocatore.fattoreVittoria = 1;
giocatore.storicoBudget = cifra_iniziale;

giocatore(2).nome = 'Pluto';
giocatore(2).budget = cifra_iniziale;
giocatore(2).posta = 1;
giocatore(2).fattoreVittoria = 36;
giocatore(2).storicoBudget = cifra_iniziale;

% iterazioni del gioco
while (siContinuaAGiocare(giocatore, banco))

    % scegliere giocata del giocatore1
    % se dispari == 0 giocatore 1 sceglie pari
    % se dispari == 1 giocatore 1 sceglie dispari
    dispari = round(rand(1));

    % giro la roulette, numero random tra 0 - 37
    % 37 equivale a 00
```

```
32      numero = giraLaRoulette();
33
34      %Calcolo del vettore della vittoria dei giocatori
35      if(numero == 37 || numero == 0)
36          vince([1, 2]) = 0;
37      else
38          if(mod(numero,2) == dispari)
39              vince(1) = 0;
40          else
41              vince(1) = 1;
42          end
43
44          if numero == 15
45              vince(2) = 1;
46          else
47              vince(2) = 0;
48          end
49      end
50
51      %Calcolo ricompense giocatori e banco
52      for ii = 1 : numel(giocatore)
53          if giocatore(ii).budget >= giocatore(ii).posta
54              if vince(ii) == 0
55                  %Sconfitta giocatore
56                  giocatore(ii).budget =  giocatore(ii).budget -
                        giocatore(ii).posta;
57                  banco =  banco + giocatore(ii).posta;
58              elseif vince(ii) == 1
59                  %Vittoria giocatore
60                  giocatore(ii).budget =  giocatore(ii).budget +
                        giocatore(ii).fattoreVittoria * giocatore(ii
                        ).posta;
61                  banco =  banco - giocatore(ii).fattoreVittoria
                        * giocatore(ii).posta;
62              end
63          end
64      end
65
66      %Aggiorno storico giocatori e banco
67      for ii = 1 : numel(giocatore)
68          giocatore(ii).storicoBudget(end + 1) = giocatore(ii).
               budget;
69      end
70      storicoBanco(end + 1) = banco;
```

```
71
72 end
73
74 plotRoulette(giocatore, storicoBanco);
```

```
1 function numero = giraLaRoulette()
2
3 numero = randi(38)-1;
```

```
1 function res = siContinuaAGiocare(giocatore, banco)
2
3 budgetCorrenti = [giocatore.budget];
4 posta = [giocatore.posta];
5 res = (any(budgetCorrenti >= posta) && banco > 0);
```

```
1 function plotRoulette(giocatore, storicoBanco)
2
3 spessore = 2;
4
5 figure();
6 plot(giocatore(1).storicoBudget , 'r' , 'LineWidth' , spessore)
7 hold on;
8 plot(giocatore(2).storicoBudget, 'b' , 'LineWidth' , spessore)
9 plot(storicoBanco , 'k' , 'LineWidth' , spessore)
10 title('Evoluzione della Roulette nel tempo');
11 xlabel('Numero giocata');
12 ylabel('Euro');
13 legend(giocatore(1).nome , giocatore(2).nome, 'Banco','Location
   ','northwest');
```

Soluzione dell'esercizio 11.9

```
1 clear
2 clc
3 close all
4
5 load scontrini
6
7 clienti = [scontrini.IDcliente];
8
9 saldo = struct('ID',[],'totale',[]);
10 n_clienti = 0;
11
```

```
for ii = 1:length(clienti)
    if (~any([saldo.ID] == clienti(ii)))
        n_clienti = n_clienti + 1;
        saldo(n_clienti).ID = clienti(ii);

        idx = [scontrini.IDcliente] == clienti(ii);
        pFagiolo = sum([scontrini(idx).punti]);
        puntiSpesa = sum(floor([scontrini(idx).totale]/10));
        saldo(n_clienti).totale = pFagiolo + puntiSpesa;
    end
end

tot_punti = [saldo.totale];
statistica(1) = sum(tot_punti <= 1000);
statistica(2) = sum(tot_punti > 1000 & tot_punti <= 5000);
statistica(3) = sum(tot_punti > 5000);
```

Soluzione dell'esercizio 11.10

```
clear
clc
close all

% acquisizione dati
dati = acquisizione_dati_meteo();

% richiesta dato da visualizzare
[city, tipo] = interrogazione_archivio_meteo();

% ricerca dati e restituzione min, media, max
[dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo);

% stampa a video delle statistiche
stampa_statistiche(dati_selezionati, city, tipo, minimo, medio,
    massimo);
```

```
function dati = acquisizione_dati_meteo()
    next = 1;
    dati = [];
    ii = 0;

    while next == 1
```

```
        ii = ii + 1;

        dati(ii).city = input('Citta'': ','s');
        dati(ii).giorno = input('Giorno: ');
        dati(ii).mese = input('Mese: ');
        dati(ii).anno = input('Anno: ');
        dati(ii).tipo = input('Tipo: ','s');
        dati(ii).valore = input('Valore: ');

        next = input('Per inserire un nuovo record premere 1,
           altrimenti 0: ');
    end

    fprintf('%d dati inseriti.\n', ii);
end
```

```
function [city, tipo] = interrogazione_archivio_meteo()
    city = input('Citta` di interesse: ', 's');
    tipo = input('Tipo misura da selezionare: ', 's');
end
```

```
function [dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo)

    for ii = 1:numel(dati)
        res(ii) = strcmp(dati(ii).city,city);
    end

    indici = res & [dati.tipo] == tipo;

    dati_selezionati = dati(indici);

    minimo = min([dati_selezionati.valore]);
    massimo = max([dati_selezionati.valore]);
    medio = mean([dati_selezionati.valore]);
end
```

```
function stampa_statistiche(dati_selezionati, city, tipo,
   minimo, medio, massimo)
    fprintf('Statistiche della misura %c in citta'' %s\n', tipo
       , city);

    for r = dati_selezionati
```

```
5           fprintf('%d/%d/%d %f\n', r.giorno, r.mese, r.anno, r.
               valore);
6       end
7
8       fprintf('\nMin: %3.2f, med: %3.2f, max: %3.2f\n', minimo,
           medio, massimo);
9   end
```

Soluzione dell'esercizio 11.11

```
1   clear
2   clc
3   close all
4
5   %% Inizializzazione magazzino
6   barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
7   tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
8   stock = [0 ; 300 ; 0 ; 6 ; 0 ];
9   ordine = [23 ; 100 ; 2 ; 100 ; 0 ];
10
11  %% Chiamata a ricerca
12  messaggio = ricerca(barcodes, tipo, stock, ordine, 123);
13  disp(messaggio);
14
15  %% Chiamata a ricercaMancanti
16  prodotti_non_in_stock = ricercaMancanti(barcodes, tipo, stock,
       ordine, 0);
17  disp(['Prodotti esauriti, ma in ordine: ' mat2str(
       prodotti_non_in_stock)]);
18  prodotti_non_ordinati = ricercaMancanti(barcodes, tipo, stock,
       ordine, 1);
19  disp(['Prodotti esauriti e non in ordine: ' mat2str(
       prodotti_non_ordinati)]);
20  prodotti_esauriti = ricercaMancanti(barcodes, tipo, stock,
       ordine, 2);
21  disp(['Prodotti con piu'' ordine che stock: ' mat2str(
       prodotti_esauriti)]);
22
23  %% Chiamata a aggiungiProdotti
24  new_barcode = 111;
25  new_tipo = 'X';
26  new_stock = 12;
27  new_ordine = 0;
```

```
[barcodes, tipo, stock, ordine] = aggiungiProdotto(barcodes,
   tipo, stock, ordine, new_barcode, new_tipo, new_stock,
   new_ordine)
```

```
function msg = ricerca (barcodes, tipo, stock, ordine, barcode)

bc_indici = find(barcodes == barcode);
if isempty(bc_indici)
    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
    ' non e'' in magazzino'];
else
    t = tipo(bc_indici);
    s = stock(bc_indici);
    o = ordine(bc_indici);

    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
    ' e'' di tipo ', num2str(t), '. Elementi in stock: ', ...
    num2str(s), ', in ordine: ', num2str(o) '.'];
end
```

```
function prodotti = ricercaMancanti(barcodes, tipo, stock,
   ordine, P)

switch P
    case 0 % esauriti ma in ordine
        bc_indici = find(stock == 0 & ordine > 0);
    case 1 % esauriti e non in ordine
        bc_indici = find(stock == 0 & ordine == 0);
    case 2 % prodotti con piu' ordine che stock
        bc_indici = find(ordine > stock);
end

prodotti = barcodes(bc_indici);
```

```
function [barcodes, tipo, stock, ordine] = aggiungiProdotto(
   barcodes, tipo, stock, ordine, new_barcode, new_tipo,
   new_stock, new_ordine)

barcodes = [barcodes; new_barcode];
tipo = [tipo; new_tipo];
stock = [stock; new_stock];
```

```
6 ordine = [ordine; new_ordine];
```