	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 4 Febbraio 2016		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA

Tema A

Spazio riservato ai docenti

- 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 2 ore.
- 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.
- Verranno valutate meglio le soluzioni che sfruttano le particolarità di Matlab per operare su matrici e vettori.

Esercizio 1 (7 punti)

1. Si implementi in linguaggio MATLAB una funzione *partition* che prende in ingresso un vettore **V** di valori numerici non ordinati, e restituisce in uscita un nuovo vettore **T** (composto dagli stessi elementi di **V**) e un valore numerico **m**, definiti nel modo seguente:
   - Se **V** ha lunghezza dispari, **m** è l'elemento in posizione centrale di **V** e il vettore **T** composto dagli stessi elementi di **V** disposti nel senguente modo: a sinistra di **m** ci sono tutti gli elementi di **V** minori di **m** , poi compare **m** (in tutte le sue occorrenze), quindi a destra tutti gli elementi maggiori di **m**.
     Ad esempio, dato il vettore **V** = [6 5 3 4 3 1 2], la funzione restituisce il vettore **T =** [3 3 1 2 4 6 5] e il valore centrale **m** = 4.
   - Se **V** ha lunghezza pari, il valore **m** è definito come la media dei due elementi centrali di **V**, e tutti gli elementi di **V** vengono disposti in **T** in modo tale che tutti quelli minori o uguali di **m** precedono quelli maggiori di **m**.
     Ad esempio, dato il vettore **V** = [5 6 1 2 3 4], la funzione restituisce il vettore **T =** [1 5 6 2 3 4] e il valore centrale *m* = 1.5 (cioè la media dei due valori in posizione centrale 1 e 2).

2. Si sviluppi uno script MATLAB che svolge le seguenti operazioni:
   a. Acquisisce da tastiera un vettore numerico **vett** e un intero positivo **n**;
   b. Crea un vettore **M** vuoto;
   c. Svolge **n** volte le seguenti operazioni:
      i. Invoca la funzione *partition* passando come parametro **vett**, sovrascrivendo **vett** con il vettore restituito da *partition* e memorizza nella prima posizione libera di **M** il secondo valore restituito;
      ii. Stampa il nuovo valore di **vett**;
   d. Al termine dell'esecuzione del ciclo, stampa il vettore **M**.

Soluzione

Domanda 1

```
function [T, m] = partition(V)
% La funzione PARTITION calcola, a partire da un array V, un numero m
% (pari al valore che V assume nella sua posizione centrale) e un nuovo
% array T ottenuto da V muovendo a sinistra di m gli elementi di V minori
% di m, e muovendo a destra di m gli elementi di V maggiori di m.
% Nel caso in cui V abbia lunghezza pari, m e' la media dei due valori
% nelle posizioni centrali.
% Parametri di input
% V: vettore di valori numerici
% Parametri di output
% T: vettore ottenuto applicando a V la trasformazione descritta sopra
% m: elemento in posizione centrale di V

lung = length(V);
if lung == 0
   T = V;
   m = 0;
else
   if mod(lung, 2) == 0
      m = mean([V(lung/2), V(lung/2+1)]);
   else
      m = V(ceil(lung/2));
   end
   T = [V(V < m) V(V == m) V(V > m)];
end
end
```

Domanda 2

```
vett = input('Inserisci un vettore numerico: ');
n = input('Inserisci un valore intero positivo: ');

M = [];

for k = 1 : n
   [vett, M(k)] = partition(vett);
   disp(vett);
end

disp('Sequenza dei valori centrali: ');
disp(M);
```

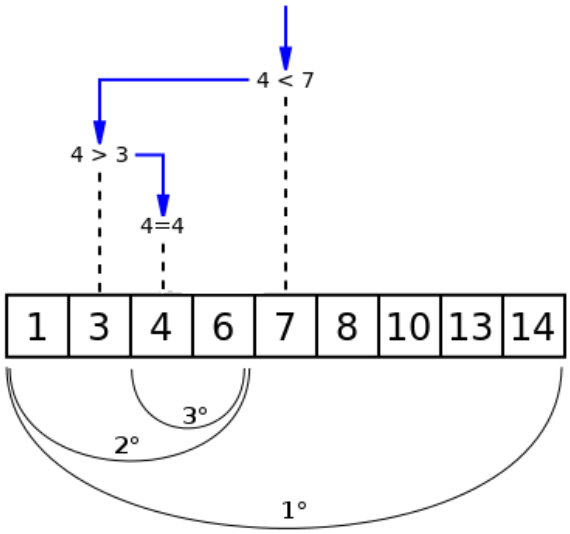
Esercizio 2 (6 punti)

La *ricerca binaria* è un algoritmo che permette di determinare se un certo valore è presente in un vettore ordinato.

La ricerca binaria opera come segue. Dato un numero N da cercare e un vettore **V** contenente valori ordinati in modo crescente e senza duplicati, si individua l'elemento nella posizione centrale di **V**; se il valore di tale elemento è uguale ad **N**, la ricerca è conclusa. Altrimenti, se questo valore è minore del numero cercato, si contiuna la ricerca nel sotto-vettore corrispondente alla metà superiore del vettore **V** (elemento centrale escluso); se invece il valore dell'elemento centrale di **V** è maggiore di **N**, si contiuna la ricerca di **N** nel sotto-vettore corrispondente alla metà inferiore del vettore **V** (elemento centrale escluso).

Questo processo va ripetuto su ciascuno dei sotto-vettori identificati finché il numero N non viene trovato, oppure tutti i valori del vettore **V** sono stati scartati.

Lo schema seguente mostra, ad esempio, la ricerca binaria del numero 4 nel vettore [1, 3, 4, 6, 7, 8, 10, 13, 14].



1. Scrivere in linguaggio MATLAB una funzione ricorsiva *ricercaBinaria* che implementi l'algoritmo di ricerca binaria descritto. La funzione deve ricevere in input come primo parametro il numero **N** da cercare e come secondo parametro il vettore riga **V** contenente i valori ordinati in modo crescente; la funzione deve restituire un valore logico **res** (vero / falso) che rappresenti il risultato della ricerca;

2. Si invochi la funzione *ricercaBinaria* con i seguenti parametri: 19 e [1, 4, 5, 7, 9, 10, 12, 15, 16, 18, 20], si indichi, motivando la risposta, quante chiamate della funzione vengono eseguite prima di ottenere il risultato. Si riporti ogni chiamata della funzione *ricercaBinaria* con i relativi parametri attuali.

Soluzione

Domanda 1

```
function [trovato] = ricercaBinaria(N, V)

% La funzione ricercaBinaria esegue la ricerca di un elemento in un
% vettore ordinato.
% La funzione suppone che il vettore sia ordinato in ordine crescente e
% adotta l'algoritmo di ricerca binaria.
%
% Parametri di input:
% N: numero da cercare
% V: vettore di valori numerici ordinati in modo crescente
%
% Parametri di output:
% trovato: valore logico pari a 1 se N si trova in V, 0 altrimenti.

lung = length(V);
centro = round(lung/2);
if lung == 0
        trovato = false;
elseif V(centro) == N
        trovato = true;
elseif V(centro) > N
        trovato = ricercaBinaria(N, V(1:(centro - 1)));
else
        trovato = ricercaBinaria(N, V((centro + 1):end));
end
```

Domanda 2

Come si nota dalla sequenza di chiamate mostrate qui sotto con i relativi parametri attuali , sono necessarie cinque chiamate alla funzione per verificare che l'elemento dato non è presente nel vettore dato.

Chiamata 1 *ricercaBinaria(*19, [1, 4, 5, 7, 9, 10, 12, 15, 16, 18, 20])

Chiamata 2 *ricercaBinaria(*19, [12, 15, 16, 18, 20])

Chiamata 3 *ricercaBinaria(*19, [18, 20])

Chiamata 4 ricercaBinaria(19, [20])

Chiamata 5 *ricercaBinaria(*19, [])

Esercizio 3 (4 punti)

A) Si consideri il seguente script in linguaggio MATLAB. Si dica quante e quali linee del codice corrispondono ad un'interazione con una periferica (si utilizzino i numeri di riga riportati di fianco al testo). Si assuma che la funzione *calcolaCosto* utilizzata dallo script non interagisca con alcuna periferica.

```
1.  load log.mat energiaProdotta energiaConsumata; % carica le variabili energiaProdotta energiaConsumata
2.  prezzoEnergia = input('Inserire il prezzo energia: ');

3.  x = 1:size(energiaProdotta, 2);

4.  ylabel('Energia prodotta');
5.  xlabel('Minuto');
6.  title('Produzione energia');
7.  plot(x, energiaProdotta);

8.  indici = find(energiaProdotta >= energiaConsumata);
9.  esubero = energiaProdotta(indici);

10. [spesaTotale, media] = calcolaCosto(energiaProdotta, energiaConsumata, prezzoEnergia);

11. disp(['Spesa totale per l''energia elettrica: ', num2str(spesaTotale)]);
12. disp(['I punti di esubero sono: ', num2str(esubero)]);
13. disp('La media di energia elettrica prodotta e la media di quella consumata sono: ');
14. disp(num2str(media));
```

B) Si supponga che su un calcolatore monoprocessore ci siano due processi pronti (P2 e P3), un processo in attesa (P4), e uno in esecuzione (P1). Si supponga inoltre che P1 esegua il codice dello script riportato sopra. Si spieghi che cosa succede quando P1 è in esecuzione e l'unità di elaborazione esegue la prima istruzione di interazione con una periferica..

C) Si supponga che la memoria fisica del calcolatore abbia le seguenti caratteristiche:

- Dimensione totale 1 GB;
- Dimensione di ciascuna pagina 64 KB.

Si faccia sempre riferimento ai quattro processi P1, P2, P3, e P4 e si supponga che ciascuno di essi utilizzi 3 pagine di memoria. Si supponga inoltre che il sistema operativo del calcolatore utilizzi 2 pagine di memoria.
Quante pagine di memoria fisica sono occupate in questo caso e quante rimangono libere, a disposizione di altri processi? Si motivi la risposta.
Si supponga ora che ciascuno dei quattro processi usi una delle tre pagine di memoria per il codice e le rimanenti due per i dati, e si supponga inoltre che P1 e P2 condividano lo stesso codice (questi processi quindi sono due diverse esecuzioni dello stesso programma), quante pagine di memoria fisica risulterebbero effettivamente occupate? Si motivi la risposta.

Soluzione

A) Si hanno 10 interazioni con periferiche nei punti 1, 2, 4, 5, 6, 7, 11, 12, 13, 14

B) Quando l'unità di elaborazione esegue l'istruzione al punto 1 dello script, si verifica un'interruzione interna che viene gestita dal sistema operativo (supervisor call). Il sistema operativo sposta P1 in attesa e manda in esecuzione un processo, scegliendo tra P2 e P3, a seconda della loro posizione nella coda dei processi pronti. Supponiamo che P2 sia il primo processo in coda. In questo caso, nella nuova situazione ci sarà un solo processo in stato di pronto (P3), un processo in stato di esecuzione (P2) e due processi in stato di attesa (P1 e P4).

C) 1 GB corrisponde a 2^{30} byte. 64 KB corrispondono a 2^{16} byte. Svolgendo la divisione $2^{30} / 2^{16}$ si ottiene che il numero di pagine in memoria è pari a 2^{14}. Il numero totale di pagine occupate dai quattro processi e dal sistema operativo sarà pari a 4 * 3 + 2 = 14. Rimarranno quindi a disposizione per altri processi 2^{14} - 14 pagine.

   Nel caso P1 e P2 condividessero lo stesso codice, allora P3 e P4, come nel caso precedente, occuperebbero 3 pagine ciascuno, mentre P1 e P2 ne occuperebbero solo 5 in tutto (2 ciascuno per i dati e una in comune per il codice). Aggiungendo a queste le due pagine occupate dal sistema operativo si avrebbe: 2 * 3 + 5 + 2 = 13.