	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Appello del 20 Febbraio 2014 Prof Giacomo Boracchi		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 3 esercizi e deve essere debitamente compilato con cognome e nome, numero di matricola, posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di 1 ora e 30 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!) **facendo in modo 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 permesso 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 meno le soluzioni che NON sfruttano i vantaggi sintattici del linguaggio Matlab/Octave nella gestione degli array.**

Esercizio 1

Si considerino le seguenti dichiarazioni di tipi e variabili che definiscono le informazioni relative ai laptop presenti nel catalogo di un distributore di informatica

```
typedef char Stringa[100];
typedef struct{
    int gb;
    int rpm; // round per minutes
}Hdd;
typedef struct{
    int mHz;
    int gb;
    Stringa tipologia;
}Ram;
typedef struct{
    int ghz;
    Ram ram[4];//ogni elemento racchiude i dati relativi ad un banco di ram
    int nBanchiRam; // dimensioni effettive dell'array ram
    Hdd hardDisk;
    Stringa produttore;
}Laptop;
```

1. Si dichiari una variabile **`listino`** atta a contenere le informazioni relative ai vari laptop in catalogo e si assuma che questa venga popolata opportunamente (il catalogo contiene al massimo 30 laptops). Si assuma esista una variabile **`n`** che al termine di tale operazione indichi il numero di laptop effettivamente inseriti in **`listino`**.

2. Si scriva un frammento di codice, che includa la dichiarazione di eventuali variabili aggiuntive, per identificare il miglior laptop del listino. Il miglior laptop è quello dotato del maggior numero di Gb di ram (inteso come la somma delle dimensioni dei banchi di ram presenti) e, a parità di ram, dall'hard disk con il valore più elevato di round per minutes (rpm). A parità degli elementi sopra due laptop sono da considerarsi equivalenti e basta stamparne uno solo. Relativamente al miglior laptop, si stampi a schermo nome del produttore, ram totale e numero di rpm dell'hard disk.

3. Si dichiari una seconda variabile **`selezione`** atta a contenere tutti i laptop di una determinata marca e vi si copi all'interno, senza lasciare buchi, tutti i laptop dello stesso produttore del laptop ritenuto migliore (nel caso in cui non si fosse sviluppato il punto precedente si assuma che il laptop migliore sia stato salvato in una variabile **`best`**).

4. Si modifichino i tipi di dato sopra definiti (definendone eventualmente di aggiuntivi) per poter includere nella descrizione di ogni portatile:
    a. La memoria cache, che è caratterizzata da tipologia (L1 o L2), da tutti gli indicatori che ne determinano il tempo di accesso medio, e dalla dimensione.
    b. La risoluzione massima dello schermo (es 1440 x 900)

    Si privilegino scelte che facilitano l'accesso a queste informazioni e la comprensione del programma.

Soluzione

```
#include <stdio.h>
#define N 30

typedef char Stringa[100];
typedef struct{
    int gb;
    int rpm; // round per minutes
}Hdd;

typedef struct{
    int mHz;
    int gb;
    Stringa tipologia;
}Ram;

typedef struct{
    int ghz;
    Ram ram[4]; // ciascun elemento indica i Gb di un banco di ram
    int nBanchiRam; // dimensioni effettive dell'array ram
    Hdd hardDisk;
    Stringa produttore;
}Laptop;

int main ()
{

Laptop listino[N],selezione[N];
int n; // numero di laptop effettivamente presenti in listino

/*  porzione di codice mancante  in cui vengono popolate la variabili listino e n*/

Laptop best;
int i, j, t, dimRam, ramMax;

// si assuma che listino e n siano stati correttamente popolati

best = listino[0];
ramMax = 0;

for(j = 0; j < best.nBanchiRam; j++)
        ramMax+=best.ram[j].gb;

for(i = 0; i < n; i++)
{
    dimRam = 0;
    for(j = 0; j < listino[i].nBanchiRam; j++)
        dimRam+=listino[i].ram[j].gb;
    if (dimRam > ramMax || (dimRam == ramMax) && listino[i].hardDisk.rpm >
best.hardDisk.rpm)
        {
            best = listino[i];
            ramMax = dimRam;
        }
}
printf("\nil laptop migliore è di marca %s ha %d Gb di ram e hdd con %d rpm",
best.produttore, ramMax, best.hardDisk.rpm);
```

```
j = 0;
for(i = 0; i < n; i++)
{
    if(strcmp(listino[i].produttore, best.produttore) == 0)
        {
            selezione[j] = listino[i];
            j++;
        }
}

for(i = 0; i < j; i++)
{
    ramMax = 0;
    for(t = 0; t < selezione[i].nBanchiRam; t++)
        ramMax += selezione[i].ram[t].gb;

        printf("\nil laptop di marca %s ha %d Gb di ram e hdd con %d rpm",
selezione[i].produttore, ramMax, selezione[i].hardDisk.rpm);
}

}


typedef struct{
    int mb;
    Stringa tipologia;
    float ht; // hit time
    float mp; // miss penality
    float hr; // hit rate
}Cache;

typedef struct{
    int risoluzioneX; // risoluzione orizzontale
    int risoluzioneY; // risoluzione verticale
}Schermo;


typedef struct{
    int ghz;
    Ram ram[4]; // ciascun elemento indica i Gb di un banco di ram
    int nBanchiRam; // dimensioni effettive dell'array ram
    Hdd hardDisk;
    Stringa produttore;
    Cache   cache;
    Schermo schermo;
}Laptop;
```

Esercizio 2

Si assuma di avere una matrice di 2 righe ed un numero arbitrario di colonne, ad esempio:

$$M = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 4 & 3 \end{bmatrix}$$

Ogni colonna della matrice rappresenta una frazione (ad esempio, per M):

$$\begin{bmatrix} \frac{1}{2} & \frac{3}{4} & \frac{1}{3} \end{bmatrix}$$

Si scriva una funzione **ricorsiva frac** che, ricevuta una qualsiasi matrice $\boldsymbol{M}$ in ingresso, ritorni due parametri $\boldsymbol{n}$ e $\boldsymbol{d}$ così definiti: $\boldsymbol{n}$ rappresenta il numeratore ed $\boldsymbol{d}$ il denominatore della frazione risultante dalla somma delle frazioni contenute in $\boldsymbol{M}$.

Ad esempio:

```
[n d] = frac([1 3 1; 2 4 3])
```

restituisce

```
n = 19
d = 12
```

poichè:

$$\frac{1}{2} + \frac{3}{4} + \frac{1}{3} = \frac{12 \times 1 + 6 \times 3 + 2 \times 4}{2 \times 4 \times 3} = \frac{19}{12}$$

La frazione $\frac{n}{d}$ deve essere minimizzata, ovvero non vi devono essere divisori comuni tra $\boldsymbol{n}$ e $\boldsymbol{d}$. Nel progettare **frac**, si può utilizzare la funzione $\mathbf{gcd}(\boldsymbol{a}, \boldsymbol{b})$ che restituisce il massimo comun divisore tra $\boldsymbol{a}$ e $\boldsymbol{b}$ per minimizzare una frazione:

$$\frac{a}{b} = \frac{\frac{a}{\gcd(a,b)}}{\frac{b}{\gcd(a,b)}}$$

Spazio soluzione

```
function [n, d] = frac(M)
   if size(M,2) == 1
      n = M(1,1);
      d = M(2,1);
   else
      [c d] = frac(M(:,2:end));
      a = M(1,1);
      b = M(2,1);

      k = a*d + c*b;
      l = b * d;

      n = k/gcd(k,l);
      d = l/gcd(k,l);
   end
end
```

Esercizio 3

Si assuma di dover convertire con un programma C degli indirizzi fisici in indirizzi virtuali. La tabella di conversione è memorizzata nel seguente array bi-dimensionale:

```
int tabella[4][2] = {
    { 0, 4 },
    { 1, 2 },
    { 2, 1 },
    { 3, 0 } };
```

dove, in ciascuna riga della tabella, viene specificato il numero di pagina fisica (NPF) ed il numero di pagina virtuale (NPV) corrispondente. Ad esempio, la prima riga di tabella dice che il numero di pagina fisica 0 deve essere tradotto in un numero di pagina virtuale pari a 4.

Si assuma che l'offset sia composto da 8 bit e si scriva un programma C che

1) richiede all'utente un indirizzo fisico e lo salva nella variabile

```
int indirizzo_fisico;
```

   Il programma quindi calcola e stampa a schermo il numero di pagina fisica corrispondente ad **`indirizzo_fisico`**

2) calcola e stampa a schermo l'indirizzo virtuale corrispondente secondo l'associazione in **`tabella`**. Si assuma che l'indirizzo abbia un NPF tra quelli presenti in **`tabella.`**

Soluzione

```
    ...
    int npf;
    int npv;
    int offset;

    offset = indirizzo_fisico % 256;
    npf    = (indirizzo_fisico - offset)/256;

    printf("Offset:      %x\n", offset);
    printf("npf:         %x\n", npf);

    for (int i = 0; i < 4; ++i) {
        if(tabella[i][0] == npf) {

            int indirizzo_virtuale;
            indirizzo_virtuale = (tabella[i][1] * 256) + offset;

            printf("Conversione: %x -> %x\n", indirizzo_fisico,
indirizzo_virtuale);
            break;
        }
    }
    return 0;
}
```