INFORMATICA - CdL in FISICA
PROVA SCRITTA DEL 02/07/2009

Scrivere **in stampatello** COGNOME, NOME e MATRICOLA su ogni foglio consegnato

N.B.: In tutti gli esercizi viene valutata anche la leggibilità del codice proposto. Inoltre, non è consentito l'uso di istruzioni che alterino il normale flusso dell'esecuzione (come, ad esempio, `continue`, `break` e istruzioni di `return` all'interno di cicli che ne provochino l'uscita forzata).
Utilizzare il tipo `boolean` definito da `typedef enum {false,true} boolean` invece di interi come variabili booleane;
In tutti gli esercizi non è consentito l'uso di variabili `static`

ESERCIZIO 1 (6 punti)
Si dice **sequenza di multipli** una sequenza di interi positivi tali che ogni intero (tranne il primo) sia multiplo del precedente. La sequenza termina quando l'utente immette un intero che viola la sequenza, cioe' che e' negativo o e' positivo ma non multiplo del precedente. Il carattere che termina la sequenza non e' considerato appartenere alla sequenza. Si legga una sequenza di multipli e si stampi la posizione del secondo intero dispari (se esiste) e la lunghezza totale della sequenza.

Stampare:

- la posizione del secondo intero dispari (se esiste);
- la lunghezza totale della sequenza;

Se ad esempio la sequenza digitata dall'utente è:

```
"-1"
```

un possibile output del programma è:

```
Mi dispiace non ci sono stati 2 numeri pari.
Lunghezza seq. : 0
```

mentre se la sequenza digitata dall'utente è:
"3 4"
un possibile output del programma e':

```
Mi dispiace non ci sono stati 2 numeri pari.
Lunghezza seq. : 1
```

Se la sequenza digitata dall'utente è:
"3 15 60 62"'
un possibile output del programma è:

```
Mi dispiace non ci sono stati 2 numeri pari.
Lunghezza seq. : 3
```

Infine se la sequenza digitata dall'utente è:
"3 15 60 180 -1"'
un possibile output del programma è:

```
Pos secondo pari:4
Lunghezza seq. : 4
```

ESERCIZIO 2 (6 punti)
Si consideri un array di dimensione `dim`, chiamato `formula` di caratteri 'F' (False **ff**), 'T (True **tt**), 'A' (And $\wedge$), 'O' (Or $\vee$) che rappresenta una formula booleana *ben formata*, con la convenzione che gli operatori 'And' e 'Or' abbiano la stessa priorità e associno a destra. Scrivere la funzione **ricorsiva** `check-ric` che, dato un array di caratteri, valuti la formula booleana data. Ad esempio se `formula` fosse il seguente array

'T'	'O'	'F'	'A'	'T'	'O'	'F'

allora la corrispondente formula booleana sarebbe ($\mathbf{tt} \vee (\mathbf{ff} \wedge (\mathbf{tt} \vee \mathbf{ff}))$) (vedi la derivazione corrispondente) e la funzione `check-ric` dovrebbe restituire `true`.

ESERCIZIO 4 (15 punti)
Si vogliono rappresentare liste che *possono* essere cicliche nel senso che l'ultimo elemento *puo'* puntare ad uno dei precedenti.

Per facilitare la manipolazione di questo tipo di liste e' ragionevole inserire un campo che dice se il puntatore al prossimo elemento punta ad un elemento precedente.

(i) (**1 punti)** Si definiscano i tipi di dato necessari per implementare in `C` la rappresentazione indicata. Si identifichi con `ListeCicliche` il tipo di dato principale.

Si definiscano le seguenti operazioni su oggetti di tipo `ListeCicliche` mediante opportune procedure o funzioni.

(ii) (**3 punti**) Dato una lista, se e' ciclica, renderla non ciclica;

(iv) (**5 punti**) Dato un medico, inserire un nuovo appuntamento;

(v) (**6 punti**) Cancellare un elemento nella lista, aggiustando l'informazione sulla ciclicita' della lista.

N.B. Le procedure/funzioni **NON** devono contenere alcuna istruzione di input/output (ad es. `scanf, printf, getchar, putchar,...`)