Esercizio 0

Scrivere un programma che legga da tastiera la dimensione n (> 0) di un array A e i suoi elementi. Si assuma che ciascun intero inserito da tastiera sia seguito da $'\backslash n'$.

```
int n, i, *A;

scanf("%d", &n);
if(n <= 0) return 1;

A = malloc(n * sizeof(int));
if(A == NULL) return 1;

for( i = 0; i < n; i++ )
     scanf("%d", A+i); // A+i <===> &A[i] entrambi denotano
                       // l'indirizzo dell'i-esimo elemento di A
```

Esercizio 1

Scrivere un programma che legga da tastiera un array A di interi senza conoscere a priori la sua lunghezza. La lettura termina appena si legge il primo "non" intero. Pseudocodice alla lavagna.

Esercizio 2

Scrivere un programma che legga da tastiera un array A di interi ordinato (in ordine non decrescente). Se A è effettivamente ordinato si deve stampare $s\ x\backslash n$ dove x è il numero di interi distinti in esso contenuti o $n\backslash n$ altrimenti.

Esercizio 3

Implementare `selection sort` e `insertion sort` e scrivere due programmi per testare le due implementazioni. Confrontare le loro prestazioni con il comando `time` utilizzando due array di 50.000 (utilizzate i due file di test che trovate sul sito del corso). Con il primo file otterrete un array con elementi generati casualmente mentre con il secondo un array già ordinato. Come mai l' `insertion sort` è migliore sul secondo array?