Esercizio 1

Si implementi `insertion sort`: si scriva un programma che legga da tastiera un array A di n interi, chiedendo n all'utente (e senza farvi assunzioni sopra). Il programma utilizza insertion sort per ordinare in modo non decrescente l'array A e lo ristampa a video, tutto su una riga.

Esercizio 2

Si implementi ora `insertion sort` su un array di stringhe. Si scriva un programma che legga da tastiera un array A di n stringhe (chiedendo n all'utente e senza farvi assunzioni sopra). Si assuma che tutte le stringhe siano piu' corte di 256 caratteri. Si ordini A usando insertion sort generando l'ordine lessicografico non decrescente (si scriva manualmente la funzione per controllare quale stringa è minore dell'altra). Si stampi infine A a video, una stringa per riga.

Esercizio 3

Si implementi `merge sort` per gli interi, ricorsivamente, iterativamente e nel metodo *misto*. Quest'ultimo è un ibrido tra merge sort ricorsivo e insertion sort: quando la dimensione della porzione di array scende sotto 20 elementi, la porzione è ordinata usando insertion sort.

Si compiano dei test di performance: si generino sequenze di interi casuali usando `rand`, di almeno 10.000 interi. Si scriva tali interi in un file tramite redirezione dell'output e si usi *time* per controllare le prestazioni dei tre algoritmi sulla stessa sequenza. Si ripeta il procedimento con molteplici sequenze (almeno 10) e si utilizzi il risultato medio per compilare una classifica tra i tre algoritmi.