Programmazione I
Prova di Programmazione – 22 gennaio 2015 – 2 ore 15 min

Partendo dal frammento di codice fornito, realizzare un programma di gestione delle *parole*, ossia delle sequenze di caratteri non separate da spazi, memorizzate in un file di testo. Assumere che la lunghezza massima di ciascuna parola sia nota a tempo di scrittura del programma. Il programma deve fornire le seguenti funzionalità.

1. **carica_parole** Carica nella memoria del programma la sequenza di parole memorizzate in un file di testo dal nome definito a tempo di scrittura del programma. L'eventuale sequenza di parole precedentemente caricata nella memoria del programma è persa ogni volta che si reinvoca questa funzionalità (all'avvio del programma tale sequenza è vuota). Si ottiene il punteggio massimo se si realizza questa funzionalità senza assumere che il numero massimo di parole nel file sia noto a tempo di scrittura del programma. Per questa funzionalità, il raggiungimento del costo computazionale minimo non è rilevante ai fini del punteggio.
2. **stampa_parole** Stampa l'ultima sequenza di parole caricata, l'una dopo l'altra e separate da spazi. Ad esempio, se si è caricata la sequenza di parole da un file contenente le seguenti due righe:
   ```
   Chi ha tempo non aspetti tempo.
   <Chi la fa l'aspetti.>
   ```
   allora stampa:
   ```
   Chi ha tempo non aspetti tempo. <Chi la fa l'aspetti.>
   ```
3. **carica_parole_2** Identica alla funzionalità al punto 1, ma con in più l'eliminazione di tutti i caratteri che non sono lettere dell'alfabeto, tranne il trattino e l'apostrofo, nonché la conversione di tutte le lettere in minuscolo (ricordarsi dell'esistenza di funzioni di libreria quali *isalpha* e *tolower*). Ad esempio, partendo dallo stesso contenuto del file di cui all'esempio al punto 2, la sequenza in memoria diverrebbe:
   ```
   chi ha tempo non aspetti tempo chi la fa l'aspetti
   ```
4. **carica_parole_3** Identica alla funzionalità al punto 3, ma con in più la memorizzazione delle parole in ordine. Ad esempio, partendo dallo stesso contenuto del file di cui all'esempio al punto 2, la sequenza in memoria diverrebbe:
   ```
   aspetti chi chi fa ha l'aspetti la non tempo tempo
   ```
5. **elimina_duplicati** Elimina eventuali parole duplicate all'interno della sequenza, assumendo che la sequenza sia ordinata alfabeticamente. Ad esempio, partendo dalla sequenza mostrata nell'esempio al punto 4, si otterrebbe:
   ```
   aspetti chi fa ha l'aspetti la non tempo
   ```
   Si ottiene il punteggio massimo se si realizza questa funzionalità con costo computazionale lineare rispetto al numero di parole nella sequenza originale.

I parametri di ingresso delle funzionalità sono solo indicativi. Gestire opportunamente le situazioni di errore, tranne l'inserimento di dati in formato errato da *stdin* e la presenza di parole troppo lunghe all'interno dei file.

REGOLE

- Si può utilizzare ogni genere di manuale o di materiale didattico di altra natura
- Per superare la prova, il programma deve essere perfettamente funzionante nelle parti 1 e 2. Il voto ottenuto in questo caso è 18.
- Ciascuna funzionalità DEVE essere implementata mediante almeno una funzione.
- Il voto massimo (almeno 30) si ottiene se
  a) il programma è perfettamente funzionante in ogni sua parte
  b) tutti i principi di ingegneria del codice visti nel corso sono stati applicati