Programmazione I
Prova di Programmazione – 2 settembre 2015 – 2 ore 15 min

Partendo dal frammento di codice fornito, realizzare un programma che simula la lettura di dati, un *blocco* alla volta, da un dispositivo di memorizzazione. Un blocco è una stringa di al più *M* caratteri. Per memorizzare i dati letti dai blocchi, il programma utilizza una sequenza di *N* oggetti dello stesso tipo, che chiamiamo per brevità *contenitori*. Sia *M* che *N* sono definiti a tempo di scrittura del programma. Il campo informazione di ciascun contenitore è in grado di contenere al più la stringa di caratteri letti da un blocco. All'avvio del programma tutti i contenitori sono *liberi*, ossia disponibili per essere utilizzati per una operazione di lettura di un blocco. Il modo in cui si effettuano le letture e si utilizzano i contenitori è spiegato in dettaglio nella descrizione delle funzionalità del programma.

1. **richiedi_lettura()** Cerca un contenitore in stato *libero*, e, se lo trova, simula la richiesta di lettura di un blocco in tale contenitore facendo passare tale contenitore dallo stato libero allo stato *attesa_lettura* (vedi prossimo punto). In caso di successo ritorna l'indice di tale contenitore.
2. **effettua_lettura(idx)** Effettua una lettura dal dispositivo, riversando il contenuto letto nel contenitore di indice **idx**, solo se la lettura è stata effettivamente precedentemente richiesta per tale contenitore, ossia se tale contenitore è in stato *attesa_lettura*. In caso affermativo, simula la lettura dal dispositivo con una lettura di una stringa di al più *M* caratteri da *stdin* (saltando gli spazi bianchi ed assumendo che l'utente non inserisca stringhe troppo lunghe), e fa passare in stato *occupato* il contenitore.
3. **libera_contenitore(idx)** Fa tornare in stato *libero* il contenitore di indice **idx**, solo se è attualmente in stato *occupato*.
4. **stampa_stato()** Stampa lo stato di ciascun contenitore, preceduto dall'indice del contenitore, e seguito dal contenuto nel caso di stato occupato. Ad esempio, se *M*=5 ed *N* =4:
   ```
   0 ATTESA_LETTURA
   1 LIBERO
   2 OCCUPATO C-3PO
   3 OCCUPATO Luke
   4 LIBERO
   ```
5. **salva_stato()** Salva lo stato della sequenza di contenitori su di un file binario.
6. **carica_stato()** Carica lo stato della sequenza di contenitori da un file binario. Lo stato precedente è perso.
7. **deframmenta()** Sposta tutti i contenitori liberi in testa alla sequenza. Ad esempio, se eseguita a partire dallo stato mostrato nell'esempio al punto 4, questa funzionalità sposterebbe i due contenitori liberi (di indice 1 e 4) nelle prime due posizioni della sequenza. Si ottiene il punteggio massimo se si implementa questa funzionalità senza creare sequenze di contenitori temporanee.

I parametri di ingresso delle funzionalità sono solo indicativi. Gestire opportunamente le situazioni di errore, tranne l'inserimento di dati in formato errato e di messaggi troppo lunghi da *stdin*.

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, 2 e 3. 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