Prova di Laboratorio di Algoritmi, 18 giugno 1998, turno 1

Il testo è composto da quattro parti. Per superare l'esame è sufficiente superare la prima parte. Le altre tre possono essere risolte in un ordine qualsiasi, essendo indipendenti le une dalle altre. Ad ogni parte deve corrispondere un solo file, contenente un programma scritto in linguaggio *C standard*; è ammesso utilizzare solamente le due librerie `stdio` e `stdlib`.

Si consiglia di leggere attentamente il testo in modo da seguirne correttamente tutte le istruzioni.

1. (file `parte1.c`, punti 0) Realizzare il tipo di dato `list` delle liste **ordinate** di interi utilizzando un'implementazione **con puntatori** e seguendo l'ordinamento crescente. Le liste possono contenere ripetizioni (necessariamente contigue) di uno stesso elemento. Il file `parte1.c` deve contenere le due seguenti funzioni:

   - `insert`: inserisce un intero `i` in una lista `l`, rispettando l'ordine. L'inserimento va fatto anche nel caso in cui `l` contenga già `i`;
   - `print`: stampa sullo standard output il contenuto di una lista `l` (in ordine crescente).

   La funzione `main` del file `parte1.c` deve realizzare (in ordine) i seguenti punti:

   - creazione di una lista vuota `l`;
   - inserimento in `l` di una sequenza di interi letti da un file di input (usando la `insert` di sopra);
   - stampa (sullo standard output) di `l` (usando la `print` di sopra).

   Il programma deve prevedere la lettura da tastiera del nome del file di input. Il file di input è una sequenza (anche vuota) di interi ≥ 0 (soliti separatori: blank, tabbing, new-line), terminata dal numero -1.

   Esempio:

   ```
   7 45 23 1 0 2 78 19 1 34
   -1
   ```

2. (file `parte2.c`, punti 1) Estendere il file `parte1.c` in modo che contenga la funzione `dispose` che distrugge una lista `l`, liberandone la memoria occupata. Variante, rispetto a quanto visto a lezione: **dopo l'invocazione `l` deve essere `NULL`**.

   Funzione `main`: come quella del file `parte1.c`, più la distruzione di `l` alla fine.

3. (file `parte3.c`, punti 2) Estendere il file `parte1.c` in modo che contenga la funzione `merge` che produce la lista ordinata ottenuta dal merge di due liste ordinate `l1` e `l2`. La funzione **non deve allocare nuova memoria**, ma bensì utilizzare le celle già disponibili di `l1` ed `l2`. **La complessità nel caso peggiore deve essere** $\Theta(|\texttt{l1}| + |\texttt{l2}|)$, dove $|\texttt{l}|$ denota la lunghezza di `l`.

   Funzione `main`: deve realizzare (in ordine) i seguenti punti:

   - creazione di due liste vuote `l1` e `l2`;
   - inserimento in `l1` di una sequenza di interi letti da un file di input (usando la `insert`);
   - inserimento in `l2` di una sequenza di interi letti dallo stesso file di input (usando la `insert`);
   - stampa (sullo standard output) del merge di `l1` e `l2` (usando la `merge` e la `print`).

Il programma deve prevedere la lettura da tastiera del nome del file di input. Il file di input è una sequenza (anche vuota) di interi ≥ 0 (corrispondente alla lista `l1`), terminata dal numero -1 e seguita da un'altra sequenza (anche vuota) di interi ≥ 0 (corrispondente alla lista `l2`).

Esempio:

```
7 45 23 1 0 2 78 19 1 34
-1
34 5 1 3 67 9 0
```

4. (file `parte4.c`, punti 2) Estendere il file `parte1.c` in modo che contenga la funzione `split` che, dato un intero $k > 0$ ed una lista ordinata `l`, produca due liste ordinate `l1` ed `l2` ottenute dividendo `l` in blocchi di k elementi contigui ed inserendo tali blocchi in modo alternato in `l1` ed `l2` (notare che l'ultimo blocco avrà un numero di elementi $\leq k$).

   Esempio:

   ```
   l = 1 2 3 4 5 6 7 8 9 10
   k = 3

   l1 = 1 2 3 7 8 9
   l2 = 4 5 6 10
   ```

   La funzione **non deve allocare nuova memoria**, ma bensì utilizzare le celle già disponibili di `l`.

   Funzione `main`: deve realizzare (in ordine) i seguenti punti:

   - creazione di una lista vuota `l`;
   - inserimento in `l` di una sequenza di interi letti da un file di input (usando la `insert`);
   - lettura di k dallo stesso file di input;
   - stampa (sullo standard output) delle due liste `l1` e `l2` ottenute dallo split di `l` per il valore di k letto (usando la `split` e la `print`).

   Il programma deve prevedere la lettura da tastiera del nome del file di input. Il file di input è una sequenza (anche vuota) di interi ≥ 0 (corrispondente alla lista `l`), terminata dal numero -1 e seguita da un intero positivo (corrispondente a k).

   Esempio:

   ```
   7 45 23 1 0 2 78 19 1 34
   -1
   5
   ```