Prova di Laboratorio di Algoritmi
4 Luglio 2000, turno 1

Il testo è composto da quattro parti. Per superare l'esame è necessario superare almeno 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 direttive.

1. (file `parte1.c`, punti 15) Realizzare il tipo di dato insiemi di interi ≥ 0, **ordinati in modo crescente**, utilizzando un'implementazione **ad array, senza ripetizioni**.

   Il file `parte1.c` deve contenere le tre seguenti funzioni C:

   - `empty`: prende un intero `n` e restituisce un insieme vuoto, utilizzando un array di dimensione `n`, allocato dinamicamente;
   - `insert`: prende un intero `i` ≥ 0 ed un insieme `s`, ed inserisce `i` in `s`;
   - `print`: stampa sullo standard output il contenuto di un insieme `s`.

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

   - legge la dimensione `n` dell'array;
   - genera un insieme vuoto `s`, di dimensione `n`, utilizzando la funzione `empty`;
   - inserisce nell'insieme `s` una sequenza di interi ≥ 0 letti da un file di input (usando la `insert` di cui sopra);
   - stampa (sullo standard output) di `s` (usando la `print` di cui 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:

   ```
   45 32 80 0 78 65 8 20 98 67
   - 1
   ```

2. (file `parte2.c`, punti 3) Estendere il file `parte1.c` in modo che contenga la funzione `delete` che prende un intero `i` ≥ 0 ed un insieme `s` e cancella il più piccolo intero `j` $>$ `i` da `s`.

   Funzione `main`: come quella del file `parte1.c` più

   - lettura di un numero intero `i` ≥ 0 da input;

- cancellazione del più piccolo intero `j` $>$ `i` da `s`;
- stampa sullo standard del nuovo insieme.

3. (file `parte3.c`, punti 5) Estendere il file `parte1.c` in modo che contenga la funzione `reverse` che prende un insieme e ne inverte l'ordine. **La funzione non deve usare strutture dati ausiliarie**, ma utilizzare solo lo spazio relativo all'insieme di input.

   Funzione `main`: come quella del file parte1.c più:

   - rovesciamento di `s`, utilizzando la funzione `reverse`;
   - stampa dell'insieme rovesciato, utilizzando la funzione `print`.

4. (file `parte4.c`, punti 7) Estendere il file `parte1.c` in modo che contenga la funzione `even_odd` che prende due insiemi `s1` ed `s2` e genera due **nuovi** insiemi, il primo contenente i numeri pari contenuti nell'unione di `s1` ed `s2`, il secondo i numeri dispari. Se la dimensione di `s1` è `n1` e la dimensione di `s2` è `n2`, allora la complessità di `even_odd` deve essere `n1 + n2`.

   Funzione `main`:

   - legge la dimensione `n` degli array;
   - genera due insiemi vuoti `s1` ed `s2`, di dimensione `n`, utilizzando la `empty`;
   - inserisce in `s1` e in `s2` una sequenza di numeri interi $\neq 0$ letti da un file di input (usando la `insert` di cui sopra);
   - stampa (sullo standard output) di `s1` e `s2` (usando la `print` di cui sopra);
   - genera i nuovi insiemi, usando la `even_odd` di cui sopra;
   - stampa i nuovi insiemi, usando la `print` di cui sopra.

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

   Esempio:

   ```
   45 32 80 0 78 65 8 20 98 67
   -1
   3 7 589 1 38 3
   -1
   ```