Prova di Laboratorio di Algoritmi
20 settembre 1999, 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 `tree` degli alberi binari di ricerca (BST), etichettati da numeri interi non negativi.

   Ricordiamo che i BST sono stati visti come modo di implementare gli insiemi; si tratta di alberi binari, non necessariamente completi, dove si distingue tra figlio destro e figlio sinistro e con la condizione:

   > per ogni nodo x
   > l'etichetta di x è strettamente maggiore dell'etichetta di ciascun nodo del sottoalbero di sinistra ed è strettamente minore dell'etichetta di ciascun nodo del sottoalbero di destra

   Si assume che **non ci siano ripetizioni**.

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

   - `empty`: funzione che crea un BST vuoto `t`;
   - `insert`: funzione che inserisce un intero `i` in un BST `t`, rispettando la proprietà dei BST;
   - `print`: stampa sullo standard output le etichette di tutti i nodi di un BST `t` in modo che siano ordinati in modo crescente (stampa in-order).

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

   - creazione di un BST vuoto `t` (usando la `empty` di cui sopra);
   - inserimento in `t` di una sequenza di interi letti da un file di input (usando la `insert` di cui sopra);
   - stampa (sullo standard output) di `t` (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:

```
8  12  4  1 34  67 0 23 1 15
-1
```

2. (file `parte2.c`, punti 3) Estendere il file `parte1.c` in modo che contenga la funzione `dispose` che distrugge un BST `t`, liberandone la memoria occupata (suggerimento: usare lo schema della DFS ricorsiva).

   Funzione `main`: come quella del file `parte1.c` più la distruzione dell'albero `t`, alla fine.

3. (file `parte3.c`, punti 5) Estendere il file `parte1.c` in modo che contenga la funzione `union` che produce il BST ottenuto dall'unione di due BST `t1` e `t2`. La funzione deve creare un nuovo BST diverso da `t1` e `t2`, in altre parole deve realizzare `t3` $\leftarrow$ `t1` $\cup$ `t2` (suggerimento: creare `t3` vuoto, visitare `t1` ed inserire in `t3` l'etichetta di ciascun nodo visitato in `t1`, poi visitare `t2` ed inserire in `t3` l'etichetta di ciascun nodo visitato in `t2`).

   La funzione `main` del file `parte3.c` deve realizzare (in ordine) i seguenti punti:

   - creazione di due BST vuoti `t1` e `t2`;
   - inserimento in `t1` di una sequenza di interi letti da un file di input;
   - inserimento in `t2` di una sequenza di interi letti da un file di input;
   - calcolo dell'unione di `t1` e `t2`, usando la funzione `union` di cui sopra;
   - stampa dell'unione di `t1` e `t2`, usando la `print`.

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

   Esempio:

```
8  12  4  1 34  67 0 23 1 15
-1
3 8 3 34 2 67 8
-1
```

4. (file `parte4.c`, punti 7) Estendere il file `parte1.c` in modo che contenga la funzione `nodes_number` che determina il numero di nodi contenuti in un BST `t`. Il numero dei nodi è definito induttivamente come segue:

   `nodes_number(vuoto) = -1` (per non avere indefinito)
   `nodes_number(foglia) = 1`
   `nodes_number(t) = 1 + nodes_number(t.left) + nodes_number(t.right)`, se `t` $\neq$ `vuoto` e `t`$\neq$ `foglia`.

   Funzione `main`:

   - creazione di un albero vuoto `t`;
   - inserimento in `t` di una sequenza di naturali letti da un file di input (usando la `insert`);
   - stampa sullo standard output del numero di nodi contenuti in `t` (usando la `nodes_number`).

   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:

```
8  12  4  1 34  67 0 23 1 15
-1
```