OII 2003

Alberi (`alberi`)

Descrizione del problema

Per gli scopi di questo esercizio, un albero è un albero binario con radice in cui tutti i nodi che non sono foglie hanno esattamente due figli e in cui tutti i nodi (comprese le foglie) sono etichettati con numeri interi non negativi. Più precisamente, un albero con N nodi avrà i nodi etichettati con i numeri da 1 a N, e nodi diversi hanno etichettediverse.

Esistono tre modi standard per enumerare le etichette dei nodi di un albero, chiamati visita anticipata, simmetrica e posticipata.

La versione ricorsiva (in pseudocodice) delle tre visite è la seguente:

```
void anticipata(T: albero);
begin
    if (T e' vuoto) return;
    stampa(etichetta di T);
    anticipata(sottoalbero sx di T);
    anticipata(sottoalbero dx di T);
end;

void simmetrica(T: albero);
begin
    if (T e' vuoto) return;
    simmetrica(sottoalbero sx di T);
    stampa(etichetta di T);
    simmetrica(sottoalbero dx di T);
end;

void posticipata(T: albero);
begin
    if (T e' vuoto) return;
    posticipata(sottoalbero sx di T);
    posticipata(sottoalbero dx di T);
    stampa(etichetta di T);
end;
```

Il problema che dovete risolvere è il seguente: date le visite anticipata e posticipata, trovare la visita simmetrica.

Dettagli di implementazione

Dovrai sottoporre esattamente un file con estensione `.c` o `.cpp`. Questo file deve implementare la funzione utilizzando il seguente prototipo.

```
void visita(int N, int *PRE, int *POST, int *SIMM )
```

Il parametro `N` è il numero di nodi dell'albero.

Gli array `PRE` e `POST` contengono la visita in preordine e postordine, l'array `SIMM` da riempire con la visita simmetrica.

Funzionamento del grader di esempio

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati di input dal file `input.txt`, a quel punto chiama la funzione che dovete implementare, e scrive il risultato restituito dalla vostra funzione sul file `output.txt`.

Nel caso vogliate generare un input, il file `input.txt` deve avere questo formato:

- Prima riga: il numero di nodi N.
- Seconda riga: N interi rappresentati la visita in preordine dell'albero .
- Terza riga: N interi rappresentati la visita in postordine dell'albero .

Assunzioni

- $1 \leq N \leq 1\,000\,000$.

Subtask

- **Subtask 1 [30 punti]:** $N \leq 1\,000$.
- **Subtask 2 [30 punti]:** $N \leq 10\,000$.
- **Subtask 3 [30 punti]:** $N \leq 100\,000$.
- **Subtask 4 [10 punti]:** nessuna limitazione specifica.

Esempio di input/output

File `input.txt`	**File** `output.txt`
7 5 3 2 1 6 7 4 2 6 7 1 3 4 5	2 3 6 1 7 5 4