Prova di Laboratorio di Algoritmi
11 febbraio 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 numeri interi ≥ 1, utilizzando un'implementazione con **tavole hash chiuse** contenenti un numero di buckets `B = 20`. Si supponga che il valore 0 venga utilizzato per rappresentare una casella vuota (cioè, l'elemento `NULL_ELEM`) e il valore -1 per rappresentare una casella dalla quale è stato cancellato un elemento (cioè, l'elemento `DELETED_ELEM`) La funzione di hash è definita come segue:

```
int h(int x)
{
 return x % B;
}
```

   La funzione di rehash è definita come segue:

```
int h(int x, int n)
{
 return (h(x) + n) % B;
}
```

   dove `n` rappresenta il numero di tentativi effettuati e `x` il valore da inserire nella tabella.

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

   - `empty`: prende una tabella e la svuota;
   - `insert`: inserisce un numero intero ≥ 1 nella tabella hash; se la tabelle è piena, visualizza un opportuno messaggio all'utente;
   - `print`: stampa sullo standard output il contenuto della tabella, bucket per bucket.

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

   - svuota una tabella `t`, utilizzando la funzione `empty`;
   - inserimento nella tabella `t` di una sequenza di interi ≥ 1 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 naturali ≥ 1 (soliti separatori: blank, tabbing, new-line), terminata dal numero -2.

Esempio:

Se il file di input contiene i seguenti valori:

```
45 32 80 78 65 8 20 98 67
-2
```

allora l'output prodotto deve essere il seguente:

```
0: 80 20
1:
2:
3:
4:
5: 45 65
6:
7: 67
8: 8
9:
10:
11:
12: 32
13:
14:
15:
16:
17:
18: 78 98
19:
20:
```

2. (file `parte2.c`, punti 3) Estendere il file `parte1.c` in modo che contenga la funzione `member` che prende un elemento `i` e una tavola `t` e restituisce vero se `i` è presente in `t`, falso altrimenti.

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

   - lettura di un numero intero ≥ 1 `i` da input;
   - determinare se `i` è presente in `t`, usando la funzione `member` di cui sopra;
   - stampare un opportuno messaggio.

3. (file `parte3.c`, punti 5) Estendere il file `parte1.c` in modo che contenga la funzione `delete` che prende un numero naturale ≥ 1 `i` e una tavola `t` e cancella `i` da `t`.

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

- lettura da tastiera numero da cancellare;
- cancellazione dell'elemento dalla tabella usando la funzione `delete`, sopra definita;
- stampa della nuova tabella.

4. (file `parte4.c`, punti 7) Estendere il file `parte1.c` in modo che contenga la funzione `union` che prende due tavole `t1` e `t2` ed aggiunge a `t1` tutti gli elementi di `t2`. Se `t1` diventa piena, viene stampato un opportuno messaggio.

   Funzione `main`:

   - inserimento in `t1` e in `t2` di una sequenza di interi letti da un file di input (usando la `insert` di cui sopra);
   - stampa (sullo standard output) di `t1` e `t2` (usando la `print` di cui sopra).
   - inserimento degli elementi di `t2` in `t1`, usando la `union` di ci sopra;
   - stampa (sullo standard output, usando la `print`), della tavola unione;

   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 naturali ≥ 1 (soliti separatori: blank, tabbing, new-line), terminata dal numero -2, seguita da una sequenza (anche vuota) di naturali ≥ 1 (soliti separatori: blank, tabbing, new-line), terminata dal numero -2.

   Esempio:

```
45 32 80 78 65 8 20 98 67
-2
7 34 6 9
-2
```