Prova scritta del 1/2/2013

Strutturare adeguatamente i programmi ed evidenziarne la strutturazione mediante indentazione. Inserire anche adeguati commenti

1) Sia `S` un tipo `struct` costituito da due campi, `a` di tipo stringa (max. *100* caratteri) e `b` di tipo numero intero. N.B. Utilizzare soltanto *stringhe "tipo C"*.

(a) Realizzare una funzione `void` di nome `aggiorna` che, presi come suoi parametri un array `A` di elementi di tipo `S`, il numero `n` di elementi in `A`, e una stringa `str`, controlla se `A` contiene già un elemento con campo `a` uguale a `str` e, in caso affermativo, incrementa di *1* il corrispondente campo `b`, altrimenti aggiunge in fondo ad `A` un nuovo elemento di tipo `S` con campo `a` uguale a `str` e campo `b` uguale a *1*. La funzione restituisce in `n` il numero di elementi presenti in `A` al termine dell'operazione.

(b) Realizzare una funzione di nome `trova` che, presi come suoi parametri un array `A` di elementi di tipo `S`, il numero `n` di elementi in `A`, un intero `k` e un array di interi `I`, memorizza in `I` gli indici di tutti gli elementi di `A` che hanno valore del campo `b` uguale a `k`. La funzione restituisce come suo risultato il numero di elementi memorizzati in `I`. Ad es., se `A` = `{{"alfa",3},{"beta",1},{"gamma",3},{"delta",7}}` e `k` = `3` allora `I` = `{0,2}`.

(c) Descrivere la funzione `aggiorna` anche tramite un diagramma di flusso.

2) Scrivere un programma principale che richiede all'utente il nome (max. *64* caratteri) di un file contenente una sequenza di stringhe separate tra loro da esattamente uno spazio, legge e memorizza il contenuto del file in un array `Testo` di elementi di tipo `S` (max. *1000* elementi), utilizzando in modo opportuno la funzione `aggiorna`. Al termine il programma chiede all'utente di fornire un intero `t` e, quindi, utilizzando (obbligatoriamente) la funzione `trova` sull'array `Testo`, determina e stampa le stringhe `a` di tutti gli elementi di `Testo` che hanno valore del campo `b` uguale a `t` (o un opportuno messaggio nel caso non ci sia alcun elemento che soddisfa). Esempi di interazione con l'utente per un file di input contenente `alfa beta alfa gamma beta delta` :

```
Dai numero: 2
Le parole presenti 2 volte sono: alfa beta
Dai numero: 3
Non ci sono parole presenti 3 volte
```

3) Sia `l` una lista concatenata i cui elementi hanno il seguente tipo:

```
struct elem
   {int info; elem* punt;}
```

(a) Dire qual è il risultato prodotto dall'esecuzione della seguente funzione `prova`:

```
void prova(elem* l) {
     if (l == NULL) return;
     do { cout << l->info << endl;
          l = l->punt;
     } while (l != NULL);
     return;
}
```

(b) Modificare la funzione `prova` in modo tale che produca lo stesso risultato, ma su una *lista circolare*, ovvero una lista concatenata in cui l'ultimo elemento punta all'elemento di testa della lista (invece che contenere `NULL`). SUGG.: La terminazione del ciclo `do-while` in questo caso si verifica quando il puntatore all'elemento successivo coincide con il puntatore di inizio lista ...).