FONDAMENTI DI INFORMATICA – modulo B (tempo 2h, punti 30/30) – 22 Giugno 2010

Esercizio 2 – PROGRAMMAZIONE IN C (punti 20)

Un file di tipo testo contiene un elenco di parole della lingua italiana (`parole.txt`), che possono comparire ciascuna nel file anche più volte. Le parole sono al massimo di 20 caratteri. Nel file ci sono almeno 10 parole distinte tra loro.
Si scriva un programma C organizzato in (almeno tre) funzioni invocate dal `main` e dedicate rispettivamente a:

1. Produrre un albero binario di ricerca T, ordinato sul campo `parola` , che riporta ogni parola una sola volta, associata al numero di occorrenze totali nel file (numero di volte che la parola compare nel file); nello svolgimento di questa funzione si richiede che il file sia scandito una sola volta; questa funzione riceve come parametri il puntatore a file, più eventuali parametri a scelta, e restituisce l'albero prodotto;

2. Caricare i primi 10 elementi dell'albero T in un vettore V (tale vettore risulterà ordinato sulla base del campo `parola`); questa funzione riceve come parametri l'albero T e il vettore V, più eventuali parametri a scelta;

3. Ordinare il vettore V sul campo `occorrenze` tramite la chiamata della funzione `qsort` ; si ricorda il prototipo della `qsort` :

   ```
   void qsort(void *base, size_t n, size_t width, int (*fcmp)(void *e1,void *e2));
   ```

   e stampare poi il contenuto del vettore V su un file di tipo testo in uscita (ORDINATE.TXT) che riporta su ogni riga una parola e a seguire sulla linea il suo numero di occorrenze (ovviamente il contenuto di tale file riporterà le prime 10 parole dell'albero T, ma in ordine di occorrenza); questa funzione riceve come parametri il vettore V e il puntatore al file di uscita, più eventuali parametri a scelta.

È possibile utilizzare ***librerie C*** (ad esempio per le stringhe). Nel caso si strutturi a moduli l'applicazione qualunque ***libreria utente*** va riportata nello svolgimento.

TEORIA – modulo B

Esercizio n. 1 (punti 6)

Dato il seguente codice in linguaggio C, dove il tipo **tree** rappresenta un albero binario di interi: e **ordins** la funzione di inserimento in un albero binario di ricerca:

```
#include <stdio.h>
#define N 10

int V[N]= {1,2,3,4,5,6,7,8,9,10};
typedef int boolean;

int common(int vet[],tree T)
{ if (T==NULL) return 0;
  else
     if (funct(T->value, vet, 0, N-1))
         return 1 + common(vet,T->left) + common(vet,T->right);
     else return common(vet,T->left) + common(vet,T->right);   }

boolean funct(int e, int V[], int i, int j)
{ int m=(i+j)/2;
  while (i<=j)
     { if (e==V[m])  return 1;
       else if (e<V[m])  j=m-1;
            else  i=m+1;
       m=(i+j)/2; }
  return 0;  }


void main(void)
{ int i;
  tree T =NULL;

  scanf("%d",&i);
  while (i>0)
       { T=ordins(i,T);
         scanf("%d",&i);    }

  printf("%d",common(V,T));
}
```

a) Si indichi cosa fa questo frammento di programma e le funzioni **funct** e **common** in particolare (non parafrasare il codice, dire in sintesi cosa fa il programma e cosa fanno le funzioni utilizzate e che tipo di algoritmo realizzano);

b) Si indichi la complessità de codice in termini di numero di test condizionali dell'istruzione **if** (test sottolineato) nella funzione **funct**, ipotizzando che da input siano stati dati **M** valori che sono stati inseriti nell'albero **T**, in funzione di **M** ed **N**. Discutere il caso migliore e il caso peggiore.

Esercizio n. 3 (punti 4)

Domanda di teoria su OOP.