FONDAMENTI DI INFORMATICA II (tempo 2h, TOT 31 punti) – 11 Marzo 2010

Esercizio n. 1 (punti 6)

Dato il seguente codice in linguaggio C, dove il tipo **`list`** rappresenta il tipo lista di interi (e la funzione **`ord_ins`** esegue inserimenti ordinati in lista):

```
list fun(int V[], int dim, list l)
{ int i; int trovato=0 ;
  list aux=NULL ;
  list root=l ;
  for (i=0 ; i<dim ; i++)
      { while ( (l!=NULL) && !trovato))
          { if (V[i]==l->value)
               { aux=ord_ins(l->value,aux);
                 trovato=1;   }
            else l=l->next;  }
        l=root;
       }
  return aux;
}
```

a) Si indichi cosa fa la funzione **`fun`** e che cosa restituisce la funzione.

b) Se ne valuti la complessità, individuando l'istruzione dominante e valutandone il numero di esecuzioni in funzione della dimensione **`dim`** del vettore **`V`** e della dimensione **`N`** della lista **`l`**.

Esercizio n. 2 (punti 21)

Due file binari, UNO.DAT e DUE.DAT, non ordinati su alcun campo, contengono i dati degli alunni di una scuola. Per ciascun alunno sono registrati il nome, il cognome, la data di nascita (tre interi, giorno mese ed anno) e il luogo di nascita. E' nota solo la dimensione del primo file (pari a 30 record), ma non quella del secondo.

Si scriva un programma C organizzato in (almeno tre) funzioni invocate dal main e dedicate a:

a) Caricare il contenuto del primo file in un vettore V ed ordinare il vettore tramite la funzione **`qsort`**; la funzione riceve (oltre ad eventuali altri parametri) il puntatore al file e il vettore V; si ricorda il prototipo della **`qsort`**:

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

b) Creare a partire dal secondo file un albero binario di ricerca T in memoria centrale; la funzione riceve (oltre ad eventuali altri parametri) il puntatore a file e restituisce il puntatore (di tipo tree) alla radice dell'albero T

Sia per il vettore sia per l'albero l'ordinamento che si vuole ottenere è in senso non decrescente in base al cognome e, a parità di cognome, in base al nome. Si suppone non vi siano omonimi.

Dal main, si chieda poi all'utente di inserire un cognome e si invochi la funzione al punto c):

c) Ricevendo in ingresso V, T e il cognome inserito dall'utente (oltre ad eventuali altri parametri), si faccia la ricerca binaria su V per controllare se esiste almeno un alunno con quel cognome e, in caso affermativo, si stampino tutti gli alunni contenuti nell'albero T che lo precedono (secondo la relazione d'ordine menzionata in precedenza)

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.

Esercizio n. 3 (punti 3)

Domanda scritta su parte OOP e Java.