FONDAMENTI DI INFORMATICA II (tempo 2h, punti 30/30) – 23 Marzo 2010

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

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

boolean fun(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",process(T));
}
```

a) Si indichi cosa fa questo frammento di programma e le funzioni **fun** e **process** in particolare (non parafrasare il codice, dire in sintesi cosa fa il programma e cosa fanno le funzioni ultilizzate);

b) Si indichi la complessità de codice in termini di numero di test condizionali dell'istruzione **if** (test sottolineato) nella funzione **fun**, 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. 2 (punti 21)

Un file binario, MY_FILE.DAT, memorizza i dati degli studenti di un corso di laurea universitario. Per ciascuno studente sono memorizzati nome e cognome (stringhe), matricola (intero), codice del corso di laurea (codice numerico come stringa di 4 char) data di nascita (nel formato giorno, mese, anno, come interi). Nel file ci sono almeno 50 studenti (non meno).
Si chiede di scrivere un programma C che ordini ed elabori il contenuto del file. In particolare, l'ordinamento che si vuole ottenere è in base al cognome, a parità di cognome in base al nome, e nel caso di omonimi in base alla matricola.
Il programma deve chiamare tre funzioni (da definire) dedicate rispettivamente a:

a) Caricare in un array V in memoria centrale i primi 50 studenti del file ed ordinare il contenuto dell'array utilizzando la funzione di libreria qsort. La funzione del punto a) riceve il puntatore al file, il vettore V, il numero N (nel nostro caso pari a 50) di elementi da caricare in V dal file (ed eventuali altri parametri a vostra scelta). Si ricorda il prototipo della funzione qsort:

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

b) Caricare il resto degli elementi contenuti nel file in memoria centrale, in una lista L ordinata. Si richiede di effettuare l'inserimento ordinato in modo iterativo. La funzione del punto b) riceve il puntatore al file, e restituisce la lista creata - un puntatore di tipo list - ed eventuali altri parametri a vostra scelta);
c) Fondere insieme il contenuto dell'array V e della lista L, scrivendo in uscita sul file ORDED_FILE.DAT. La funzione del punto c) riceve il vettore V, il numero N dei suoi elementi, la lista L e il puntatore al file di uscita (preventivamente aperto in scrittura), ed eventuali altri parametri a vostra 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.

Esercizio n. 3 (punti 3)

Domanda scritta su parte OOP e Java.

Illustrare e discutere il meccanismo di ereditarietà presente nel linguaggio Java, tra componenti classe, classe astratta e interfacce e le differenze nel caso ve ne siano.