Fondamenti di Informatica L-A (A.A. 2003/2004) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di martedì 13 luglio 2004 – durata 2h30m

ESERCIZIO 1 (14 punti)

Una software house che produce un programma di agenda elettronica salva gli appuntamenti su un file binario (al massimo 100 appuntamenti su un file). In particolare, ogni appuntamento è così strutturato:

- ora (un intero, con valore compreso tra 9 e 16 inclusi, rappresentante l'ora del giorno in cui è stato fissato un appuntamento; si supponga per semplicità che sia possibile fissare un solo appuntamento per ogni ora);
- giorno (un intero);
- mese (un intero);
- titolo (non più di 64 caratteri, senza spazi);
- descrizione (non più di 1024 caratteri, senza spazi).

Si vuole aggiungere ora una nuova funzionalità al programma di agenda: dato il file degli appuntamenti di un utente A e il file degli appuntamenti di un utente B, il programma deve restituire quali sono gli orari liberi per entrambi gli utenti, così da poter fissare una riunione tra A e B.

A tal scopo, il candidato scriva una procedura **`app_oggi()`** che riceva in ingresso un puntatore a un file binario **`f1`**, un intero rappresentante un **`giorno`**, un intero rappresentante un **`mese`**, ed un array di interi **`sched`** (di dimensione 8, con i valori tutti inizializzati a 0), rappresentante le ore della giornata lavorativa (dalle 9 alle 16 compresi). La procedura deve leggere dal file **`f1`** tutti gli appuntamenti relativi al giorno e al mese specificati, e porre a uno i valori nell'array **`sched`** agli indici corrispondenti agli orari per i quali risulta fissato un appuntamento. **(punti 7)**

Il candidato realizzi poi un programma C che chieda ad un utente un giorno ed un mese, e utilizzando la procedura **`app_oggi()`**, apra i file **`paperino.dat`** (gli appuntamenti di paperino) e **`topolino.dat`** (gli appuntamenti di topolino), e stampi a video quali sono gli orari ancora liberi contemporaneamente per paperino e per topolino. A tal scopo si creino gli array **`sched_paperino`** e **`sched_topolino`** (instanziati ad opportuni valori) e li si utilizzi per invocare la procedura **`app_oggi`**. Si tenga presente che se a un determinato orario non ci sono appuntamenti, allora il valore di **`sched`** per quell'orario deve valere 0. **(punti 7)**

ESERCIZIO 2 (6 punti)

Si scriva una funzione **`diversi()`** che date in ingresso due liste **`l1`** e **`l2`** di interi, restituisca una terza lista contenente gli interi di entrambe le liste che, occupando la stessa posizione, hanno però valore diverso. Le liste **`l1`** e **`l2`** sono non ordinate, ma hanno la stessa lunghezza. Ad esempio, se **`l1=[1,3,5,7] e l2=[1,2,5,4]`**, il risultato deve essere la lista **`[3,2,7,4]`**.
La funzione **`diversi()`** può essere realizzata in modo ricorsivo o iterativo, utilizzando il tipo di dato astratto **`list`** e le operazioni primitive sul tipo **`list`** definite durante il corso (che quindi possono NON essere riportate nella soluzione).

ESERCIZIO 3 (6 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione (si motivi opportunamente la risposta data)?

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 15

int copia( char * s ,  char * t, int * m);
int N = MAX - 7;

int main () {
    char s[] = "Angelina Jolie";
    char * t;
    int N = 13, i;

    t = malloc(N);
    N = copia(s, t, &N);
    printf("N vale adesso: %d\n",N);
    for (i=N-1; i>0; --i)
        printf("%c", t[i-1]);
    return 0;
}

int copia( char * s ,  char * t, int * m) {
    int i;
    for (i=0; i<((N<*m)?N:*m); i++)
        t[i] = s[i];
    t[i+1] = s[MAX-1];
    return i+1;
}
```

ESERCIZIO 4 (4 punti)

Si consideri la seguente funzione **FUN()**:

```
int FUN(int a, int b){
      a=a/b;
      if ((a%b)>0) return FUN(a, b) + a;
      else return 1; }
```

Si scriva il risultato della funzione quando invocata come **FUN(15, 2)** e si disegnino i corrispondenti record di attivazione.

ESERCIZIO 5 (3 punti)

Si definisca che cosa si intende per ricorsione “tail”, indicando quali vantaggi, dal punto di vista computazionale, tale tipo di ricorsione abbia rispetto alla ricorsione “non-tail”.

SOLUZIONE

ESERCIZIO 1

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 100

typedef struct {
      int ora;
      int giorno;
      int mese;
      char titolo[65];
      char descrizione[1025];
} appuntamento;

void app_oggi(FILE *f1, const int giorno, const int mese, int sched[8]) {
    int i, letti = 0;
    appuntamento impegni[DIM];

    letti = fread(impegni, sizeof(appuntamento), DIM, f1);
// bene anche lettura record per record, uno ad uno

    for (i=0; i<letti; i++) {
        if ((impegni[i].giorno == giorno) && (impegni[i].mese == mese))
            sched[impegni[i].ora-9] = 1;
    }
}

int main() {
      FILE *f1, *f2;

      int paperino_sched[8], topolino_sched[8];
      int i, giorno, mese;

      for(i=0; i<8; i++) paperino_sched[i] = 0;
      for(i=0; i<8; i++) topolino_sched[i] = 0;

    if ((f1=fopen("paperino.dat", "rb"))==NULL) {
        printf("Non sono riuscito ad aprire file1 in lettura!"); exit(1); }
    if ((f2=fopen("topolino.txt", "rb"))==NULL) {
        printf("Non sono riuscito ad aprire file2 in scrittura!"); exit(2); }

      do {  printf("Inserire giorno [gg]: ");
            scanf("%d", &giorno);
            printf("Inserire mese [mm]: ");
            scanf("%d", &mese);
      } while ((giorno >31) || (giorno<1) || (mese>12) || (mese<1));
      app_oggi(f1, giorno, mese, paperino_sched);
      fclose(f1);
      app_oggi(f2, giorno, mese, topolino_sched);
      fclose(f2);

      printf("Orari liberi per entrambi:\n");
      for(i=0; i<8; i++) {
            if ((paperino_sched[i] == 0) && (topolino_sched[i]==0))
                  printf("Orario: %d\n", i+9);
      }
      return 0;
}
```

ESERCIZIO 2

```
list diversi(list l1, list l2) {
      if (empty(l1)) return l2;
            else if (head(l1) != head(l2))
            return cons(head(l1), cons(head(l2), diversi(tail(l1), tail(l2)));
            else return diversi(tail(l1), tail(l2));
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e stampa:
N vale adesso: 9
anilegnA

Infatti la stringa s viene inizializzata ad “Angelina Jolie”, terminatore compreso. Successivamente viene allocato dinamicamente uno spazio di N byte, dove N è definito localmente alla funzione main() e quindi vale 13 (esiste un'altra variabile N definita globalmente, il cui valore è 8); l’indirizzo iniziale di tale spazio viene assegnato a t. Viene poi invocata la procedura copia() con parametri s, t e N (che vale 13). La funzione si limita a copiare alcuni valori dal vettore s al vettore t. In particolare copia i caratteri dalla posizioni 0 alla posizione determinata dal minimo valore tra il parametro m(che vale 13) e la variabile N globale (che vale 8). Vengono quindi copiati i primi 8 caratteri della stringa s nella stringa t. La funzione copia provvede poi ad aggiungere anche il carattere in posizione 15 della stringa s, che risulta essere il terminatore di stringa: quindi t è una stringa corretta.
La funzione main() si limita a stampare il valore di N modificato dall'assegnamento del risultato della funzione copia(), e poi stampa il contenuto della memoria indirizzata da t, a partire dal carattere di posizione 7 fino al carattere di posizione 0.

ESERCIZIO 4

La funzione restituisce il valore 12, attraverso la seguente sequenza di record di attivazione: