Fondamenti di Informatica L-A (A.A. 2005/2006) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Venerdì 16 Dicembre 2005 – durata 2h30m
Compito B

ESERCIZIO 1 (12 punti)

Un supermercato vuole analizzare le statistiche di spesa di alcuni suoi clienti al fine di comprenderne le abitudini. Per ogni cliente esiste un file binario su cui vengono salvate le informazioni riguardo tutte le spese che ha compiuto, tramite strutture dati ti tipo **`Shopping`**. Tale struttura contiene il nome del cliente (una stringa di al più 64 caratteri utili), il giorno in cui è stata effettuata la spesa (un intero che rappresenta il giorno nell'anno corrente), e l'importo in euro (un float). Per ogni cliente esiste un solo file; i dati di tale cliente sono salvati tutti su quel file. Il supermercato poi salva su ogni riga di un file di testo di nome "**`client.txt`**", che funge da "chiave", il nome del cliente (sempre una stringa di al più 64 caratteri utili) e, separato da uno spazio, il nome del file in cui i dati del cliente sono memorizzati (stringa di al più 64 caratteri utili). Si deve realizzare un programma che calcoli la media delle spese effettuate in un certo periodo da ogni cliente presente nel file "**`client.txt`**". A tal fine, il candidato realizzi:

```
/* Esempio di contenuto del
file "client.txt": */

PaoloBellavista abc56v.dat
PaolaMello g67hj5.dat
FedericoChesani 557fgh.dat
CarloGiannelli 88p9o2.dat
```

1. una funzione:

   `list findBills(char * fileName, int start, int end)`

   che, ricevuti in ingresso il nome del file binario e gli interi **`start`** e **`end`**, restituisca una lista contenente i soli importi delle spese effettuate in un giorno dell'anno compreso tra **`start`** e **`end`** (estremi inclusi). Si supponga a tal fine di poter disporre del tipo di dato astratto **`list`**, definito per il tipo primitivo **`float`**, e anche di tutte le funzioni primitive viste a lezione. (**5 punti**)

2. un programma **`main()`**, che chieda di inserire da input due giorni dell'anno. Per ogni cliente registrato in tale file, il programma deve stampare a video il nome del cliente e la media delle spese effettuate nel periodo indicato. A tal fine, il candidato usi la funzione definita al punto precedente per selezionare la lista delle spese fatte nel periodo specificato, e poi ne calcoli la media. (**7 punti**)

ESERCIZIO 2 (11 punti)

È dato un file di testo "**`dati.txt`**" contenente alcuni valori interi. Ogni riga del file di testo contiene una sequenza di al più 20 interi positivi, terminati sempre da uno 0; inoltre ogni riga contiene sempre almeno un numero oltre allo 0. Si deve realizzare un programma che, estratti da ogni riga gli interi (escluso il terminatore), calcoli la somma dei valori di ogni riga e salvi tali somme in una lista creata opportunamente. Il candidato definisca:

1. una funzione

   `int * readLine(FILE * fp, int * length)`

   che allochi dinamicamente memoria sufficiente per memorizzare gli interi presenti. La funzione salvi nella memoria allocata gli interi presenti su una stessa riga, fino al terminatore (escluso), e restituisca il puntatore a tale area di memoria come parametro di ritorno della funzione. Inoltre, tramite il parametro **`length`**, la funzione restituisca il numero di interi effettivamente memorizzati (in una riga ci sono al massimo 20 interi). Qualora sia stata raggiunta la fine del file **`fp`**, la funzione restituisca il valore **`NULL`**. (**5 punti**)

2. un programma **`main()`** che, aperto opportunamente il file "**`dati.txt`**", utilizzi la funzione **`readLine(…)`** per ottenere gli interi presenti su ogni riga. Per ogni insieme di interi, il programma calcoli la somma di tali valori e poi de-allochi la memoria occupata da tali valori; tutte le somme devono poi essere memorizzate in una struttura dati di tipo **`list`**. Terminata la lettura di tutte le righe, il programma stampi ogni somma calcolata e salvata nella lista, e de-allochi poi la memoria usata per gli elementi della lista. Si

supponga di possedere la definizione del tipo di dato astratto **`list`**, ma di NON possedere alcuna funzione primitiva per accedervi: il candidato quindi utilizzi direttamente la notazione tramite puntatori. (**6 punti**)

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>

typedef struct item {
      int value;
      struct item * next;
} Item;
typedef Item * list;

int dim = 4;

list split(int * p, int dim) {
      list result1=NULL, result2=NULL, temp=NULL;
      int i;
      for (i=0; i<dim; i++) {
            temp = (list) malloc(sizeof(Item));
            temp -> value = *(p+i);
            if ( p[i]%2 ==1) {
                  temp -> next = result1;
                  result1 = temp; }
            else {temp -> next = NULL;
                  result2 -> next = temp;
                  result2 = temp; }
            if (result1==NULL) result1=temp;
            if (result2==NULL) result2=temp;
      }
      return result1; }

int main() {
      int v[] = {1,2,3,4,5,6}, dim=5;
      list l1, temp;

      l1 = split(v, v[dim]);
      while (l1 != NULL) {
            printf("%d ", l1->value);
            temp = l1;
            l1 = l1->next;
            free(temp); }
      return 0; }
```

ESERCIZIO 4 (3 punti)

Il candidato illustri brevemente, aiutandosi con esempi pratici di programmazione, come si possa realizzare il passaggio di parametri per riferimento nel linguaggio C qualora si vogliano passare come parametri di invocazione ad una procedura (funzione con valore di ritorno **`void`**):

a) un float;
b) una struct;
c) un array;
d) una lista (di cui si voglia modificare il primo elemento).

SOLUZIONE

ESERCIZIO 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

#define DIM 65

typedef struct shopping {
      char name[DIM];
      int day;
      float bill;
} Shopping;

list findBills(char * fileName, int start, int end) {
      FILE * fShopping;
      list result;
      Shopping temp;

      if ((fShopping = fopen(fileName, "rb")) == NULL) {
            printf("Error opening the file %s\n", fileName);
            exit(-1); }

      result = emptylist();
      while (fread(&temp, sizeof(Shopping), 1, fShopping) > 0 )
            if (temp.day >= start && temp.day <= end)
                  result = cons(temp.bill, result);
      fclose(fShopping);
      return result; }

int main() {
      FILE * fClients;
      char fileName[DIM], clientName[DIM];
      list tempBill;
      float totalBill;
      int start, end, count=0;

      printf("Inserire data di inizio e data di fine: ");
      scanf("%d %d", &start, &end);
      if ((fClients = fopen("client.txt", "r")) == NULL) {
            printf("Error opening the file %s\n", "client.txt");
            exit(-1); }

      while (fscanf(fClients, "%s %s", clientName, fileName) != EOF ) {
            totalBill = 0;
            tempBill = findBills(fileName, start, end);
            printf("%s: ", clientName);
            while(! empty(tempBill)) {
                  totalBill = totalBill + head(tempBill);
                  count++;
                  tempBill = tail(tempBill); }
            printf(" Average bill: %6.2f\n", totalBill/count);
      }
      fclose(fClients);

      return 0;
}
```

ESERCIZIO 2

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"

#define MAX 20

int * readLine(FILE * fp, int * length) {
      int temp, i;
      int * result = NULL;

      *length = 0;
      if (fscanf(fp, "%d", &temp) != EOF) {
            result = (int *) malloc (sizeof(int) * MAX);
            while (temp != 0) {
                  result[*length] = temp;
                  *length = *length + 1;
                  fscanf(fp, "%d", &temp);
            }
      }
      return result; }

int main() {
      FILE * fp;
      int * v, length, total, i;
      list l1 = NULL, temp = NULL;


      if ((fp = fopen("dati.txt", "r")) == NULL) {
            printf("Error opening the file %s\n", "dati.txt");
            exit(-1); }

      while((v=readLine(fp, &length)) != NULL) {
            total = 0;
            for (i=0; i<length; i++) total = total + v[i];
            free(v);
            temp = (list) malloc(sizeof(item));
            temp ->value = total;
            temp->next = l1;
            l1 = temp;
      }

      fclose(fp);

      while (l1 != NULL) {
            temp = l1;
            printf("%d\n", l1->value);
            l1 = l1->next;
            free(temp);
      }
      return 0; }
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

```
5 3 1 2 4 6
```

La funzione `split(…)` controlla, a partire dal primo valore fino all'ultimo, i valori memorizzati nell'array di interi `p`. Se i valori sono dispari, allora la funzione copia tali valori nella lista `result1`, ponendoli in testa alla lista; se invece i valori sono pari, la funzione li pone in coda (a tal fine tiene traccia dell'ultimo elemento della lista tramite `result2`).

Il programma `main()` si limita a definire un array di interi `v`, e a passarlo come parametro alla funzione `split(…)`. Da notare che come dimensione dell'array viene passato il valore di `v` che corrisponde all'indice di valore `dim`, variabile ridefinita localmente (esiste una variabile di nome `dim` e di scope globale, che però non viene "vista" dentro al `main()`). Siccome `dim` vale `5`, `v[5]` vale `6`, e quindi `split` è invocato con parametri `split(v, 6)`. Infine il programma `main()` stampa a video il contenuto della lista e de-alloca opportunamente la memoria occupata.