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 A

ESERCIZIO 1 (12 punti)

Una ditta di impianti elettrici tiene un registro dei lavori effettuati per i diversi clienti su diversi file binari, e in un file di testo registra per ogni cliente il nome del file in cui sono registrati i lavori che lo riguardano. In particolare, per ogni lavoro effettuato la ditta salva su un file binario strutture dati di tipo **`Work`**: ogni struttura contiene il nome del cliente (una stringa di al più 64 caratteri utili), il giorno in cui il lavoro è stato fatto (un intero che rappresenta il giorno nell'anno corrente), e l'importo ai fini della fatturazione (un float). I lavori relativi ad uno stesso cliente risiedono tutti nello stesso file, ma uno stesso file contiene i dati dei lavori relativi a più clienti. La ditta salva poi, su ogni riga di un file di testo 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 lavori del cliente sono salvati (stringa di al più 64 caratteri utili). Si deve realizzare un programma che calcoli l'ammontare delle fatture per tutti i clienti della ditta. A tal fine, il candidato realizzi:

```
/* Esempio di contenuto del
file di testo: */

PaoloBellavista marzo.dat
PaolaMello aprile.dat
FedericoChesani marzo.dat
CarloGiannelli marzo.dat
```

1. una funzione:

   ```
   list findBills(char * fileName, char * clientName)
   ```

   che, ricevuti in ingresso il nome del file binario e il nome del cliente, restituisca una lista contenente i soli importi relativi ai lavori eseguiti per il cliente specificato (il candidato presti attenzione al fatto che in uno stesso file ci possono essere i dati relativi a più clienti). Si supponga a tal fine di poter disporre del tipo di dato astratto **`list`** visto a lezione, definito per il tipo primitivo **`float`**, e si supponga di disporre anche di tutte le funzioni primitive presentate a lezione. Al fine di confrontare le stringhe, si ricorda inoltre al candidato l'esistenza della funzione di libreria **`strcmp(…)`**. (**5 punti**)

2. un programma **`main`**, che chieda all'utente il nome del file di testo in cui sono registrate le coppie nome cliente-nome file; per ogni cliente registrato in tale file, il programma deve stampare a video il nome del cliente, la lista degli importi relativi, e la loro somma. Al fine di stampare le liste, il candidato ipotizzi di avere a disposizione la funzione **`showlist(…)`** opportunamente modificata, e che quindi non deve essere riportata nella soluzione. (**7 punti**)

ESERCIZIO 2 (11 punti)

È dato un file di testo "**`dati.txt`**" contenente alcuni interi. In particolare, in ogni riga del file di testo, il primo intero (sempre positivo e maggiore di 0) indica quanti interi vi sono sulla stessa riga successivamente. Si deve realizzare un programma che, estratti da ogni riga gli interi (escluso il primo che ha la sola funzione di indicare quanti interi vi siano successivamente), calcoli la media di tali valori e salvi le medie in una lista creata opportunamente. Il candidato definisca:

1. una funzione

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

   che, letto il primo intero, allochi dinamicamente memoria sufficiente per memorizzare gli interi presenti. La funzione salvi nella memoria allocata gli interi successivi al primo valore, 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. Qualora sia stata raggiunta la fine del file **`fp`**, la funzione restituisca il valore **`NULL`**. (**4 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 media aritmetica di tali valori e poi de-allochi la memoria occupata da tali valori; tutte le medie devono poi essere memorizzate tramite una struttura dati di tipo **`list`**. Terminata la lettura di tutte le righe, il

programma stampi ogni valore medio e quanti elementi sono presenti nella lista (tale valore deve essere pari al numero di righe presenti nel file iniziale), e de-allochi 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. (**7 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 = 5;

list split(int * p, int dim) {
      list result1=NULL, result2=NULL, temp=NULL;
      for (; dim>0; dim--) {
            temp = (list) malloc(sizeof(Item));
            temp -> value = p[dim-1];
            if ( (*(p+dim-1)) %2 ==0) {
                  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};
      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 le differenze, anche in termini di occupazione di spazio disco, fra l'utilizzo di un file di testo o di un file binario per la memorizzazione di una sequenza di dati numerici di tipo int.

SOLUZIONE

ESERCIZIO 1

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

#define DIM 65

typedef struct work {
      char name[DIM];
      int day;
      float bill;
} Work;

list findBills(char * fileName, char * clientName) {
      FILE * fWorks;
      list result;
      Work temp;

      if ((fWorks = fopen(fileName, "rb")) == NULL) {
            printf("Error opening the file %s\n", fileName);
            exit(-1); }
      result = emptylist();
      while (fread(&temp, sizeof(Work), 1, fWorks) > 0 )
            if (strcmp(temp.name, clientName) == 0)
                  result = cons(temp.bill, result);
      fclose(fWorks);
      return result;
}

int main() {
      FILE * fClients;
      char fileClients[DIM], clientName[DIM], fileName[DIM];
      list tempBill;
      float totalBill;

      printf("Inserire il nome del file dei clienti: ");
      scanf("%s", fileClients);

      if ((fClients = fopen(fileClients, "r")) == NULL) {
            printf("Error opening the file %s\n", fileClients);
            exit(-1); }
      while (fscanf(fClients, "%s %s",clientName, fileName) != EOF ) {
            totalBill = 0;
            tempBill = findBills(fileName, clientName);
            printf("%s: ", clientName);
            showlist(tempBill);
            while(! empty(tempBill)) {
                  totalBill = totalBill + head(tempBill);
                  tempBill = tail(tempBill);
            }
            printf(" Total bill: %6.2f\n", totalBill);
      }
      fclose(fClients);
      return 0;
}
```

ESERCIZIO 2

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

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

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

int main() {
      FILE * fp;
      int * v, length, i, total;
      list average = 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 /((float) length);
            temp->next = average;
            average = temp;
      }
      fclose(fp);

      i=0;
      while (average != NULL) {
            temp = average;
            printf("%f\n", average->value);
            average = average->next;
            free(temp);
            i++;
      }

      printf("Righe presenti nel file: %d\n", i);
      return 0;
}
```

ESERCIZIO 3

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

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

La funzione `split(…)` controlla, a partire dall'ultimo valore fino al primo, i valori memorizzati nell'array di interi `p`. Se i valori sono pari, allora la funzione copia tali valori nella lista `result1`, ponendoli in testa alla lista; se invece i valori sono dispari, la funzione li pone in coda alla lista (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 definita globalmente. 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.