Fondamenti di Informatica L-A (a.a. 2007/2008 e precedenti) - Ingegneria Informatica
Prof.ssa Mello
Prova d'Esame di Giovedì 17 Settembre 2009 – durata 2.30h
Totale 32 punti

ESERCIZIO 1 (12 punti)

Un laboratoro di analisi dei materiali è specializzato nella misura di precisione della resistenza elettrica dei campioni sottoposti ad indagine. L'addetto collega due conduttori elettrici ad ogni campione di materiale, vi invia degli impulsi di corrente e legge su un opportuno display i valori di resistenza elettrica minima e massima offerta dal campione. Quindi annota in un file di testo tali valori. Il file di testo, denominato "`misure.txt`", è così organizzato: su una riga l'impiegato annota il **codice** del campione sotto analisi (al più 8 caratteri alfanumerici, senza spazi); nella riga successiva l'impiegato annota il valore di resistenza minimo **min** (un float); nella riga successiva ancora l'impiegato annota il valore di resistenza massimo **max** (un float). Tale schema di annotazione si ripete per tutti i campioni sottoposti ad analisi. Purtroppo a volte l'impiegato agisce in maniera confusa, rieseguendo il test su un campione già testato in precedenza; nel file vengono annotate più misure diverse relative allo stesso campione (cioè allo stesso codice identificativo).

Il candidato definisca un'opportuna struttura dati **misura**, al fine di rappresentare i dati relativi ad una certa misurazione effettuata. Quindi si realizzi una funzione:

```
misura * leggiMisurazioni(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga i dati relativi a tutte le misurazioni e restituisca un array allocato dinamicamente di strutture dati **misura**. Non è noto a priori quante misurazioni siano registrate nel file.

Il candidato realizzi poi un programma main che usando la funzione di cui sopra legga tutte le misurazioni effettuate. Il programma abbia poi cura di stampare a video tutte le misurazioni lette, senza ripetizioni. In caso uno stesso campione (lo stesso codice identificativo) compaia più volte, il programma stampi la prima misurazione registrata nel file, ed ignori le successive.

ESERCIZIO 2 (10 punti)

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice compare quindi ripetuto più volte. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere uguale. Invece ogni giorno ci sono dei problemi.
Si scriva una funzione ricorsiva:

```
list estrai(list l1, list l2, list l3);
```

che riceve in ingresso:

- una lista `l1` di interi, rappresentanti i codici identificativi di tutti gli impiegati (codici non ripetuti);
- una lista `l2` che contiene i codici degli impiegati che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- una lista `l3` che contiene i codici degli impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);

La funzione `estrai(…)` deve restituire in uscita una lista dei codici corrispondenti alle persone che effettivamente son venuti in ufficio, e che hanno un numero di entrate/uscite discrepante.

Ad esempio, supponendo **l1=[3,7,9]**, **l2=[7,7,9,7,9]**, **l3=[7,9,9,9,7,7]**, la lista restituita come risultato sarà **[9]**, poiché l'impiegato identificato dal codice **3** non è venuto al lavoro (non è presente né in **l2**, né in **l3**): l'impiegato identificato dal codice **7** è a posto (tre ingressi in **l2**, tre uscite in **l3**); invece l'impiegato identificato dal codice **9** ha due ingressi e tre uscite.

Al fine di realizzare la funzione di cui sopra, il candidato implementi una funzione iterativa:

```
int conta(element el, list l1);
```

che restituisce il numero di volte che l'elemento **el** è presente nella lista **l1**.

ESERCIZIO 3 (4 punti)

Dati i numeri decimali **A=62**, e **B=-5**, si determini il numero minimo di bit necessari per rappresentare contemporaneamente i suddetti numeri, assumendo che il bit più significativo sia usato per rappresentare il segno. Utilizzando poi lo stesso numero di bit, si esegua l'operazione:

```
C = A-B
```

e si discuta se il risultato ottenuto è o no significativo, fornendo una opportuna spiegazione.

ESERCIZIO 4 (3 punti)

Si introduca brevemente la struttura dati dinamica "stack" vista a lezione, descrivendone le funzioni primitive di accesso e la loro implementazione.

ESERCIZIO 5 (3 punti)

Data la funzione:

```
float g(float a, float b){
      if ( (a==b) || (a<=0))
            return a;
      else
            return 1 + g(a-2, b) + g(a-1, b-0.5);
}
```

mostrare la sequenza dei record di attivazione nel caso in cui la funzione sia invocata con parametri attuali (2.0, 1.0) e il valore di ritorno.

Soluzioni

ESERCIZIO 1

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

typedef struct {
	char codice[9];
	float max;
	float min;
} misura;

misura * leggiMisurazioni(char * fileName, int * dim) {
	misura * result;
	misura temp;
	FILE * fp;
	int i;

	if ((fp=fopen(fileName, "r")) == NULL) {
		printf("A problem occurred while opening the file\n");
		exit(-1);
	}
	*dim=0;
	while (fscanf(fp, "%s%f%f", temp.codice, &(temp.min), &(temp.max)) == 3)
		(*dim)++;
	rewind(fp);
	result = (misura *) malloc(sizeof(misura)* (*dim));
	for (i=0;i < *dim;i++)
		fscanf(fp, "%s%f%f", result[i].codice, &(result[i].min), &(result[i].max));
	fclose(fp);
	return result;
}

void main() {
	misura * m;
	int dim;
	int presente;
	int i, j;
	m = leggiMisurazioni("misure.txt", &dim);
	for (i=0; i<dim; i++) {
		presente = 0;
		for (j=0; j<i && !presente; j++)
			if (strcmp(m[i].codice,m[j].codice)==0)
				presente = 1;
		if (!presente)
			printf("Misurazione: %s %f %f\n", m[i].codice, m[i].min, m[i].max);
	}
	system("PAUSE");
}
```

ESERCIZIO 2

```
int conta(element el, list l1) {
	int result = 0;
	while (!empty(l1)) {
		if (el==head(l1))
			result++;
		l1=tail(l1);
	}
	return result;
}

list estrai(list l1, list l2, list l3) {
	if (empty(l1))
		return l1;
```

```
        else {
                if (conta(head(l1), l2) != conta(head(l1), l3))
                        return cons(head(l1), estrai(tail(l1), l2, l3));
                else return estrai(tail(l1), l2, l3);
        }
}
```

ESERCIZIO 3

Il maggiore dei due numeri, in valore assoluto, è A, che vale 62. Per rappresentare il valore assoluto di A sono sufficienti 6 bit (2^6=64, [0..63]). Al fine di rappresentare anche i numeri negativi, si usa un ulteriore bit per il segno. Perciò il numero minimo di bit, necessari e sufficienti per rappresentare A e B è 7.

```
 62 ->     0111110                  0111110 ( 62) -
  5 ->     0000101                  0111011 ( -5) =
 -5 ->     1000101                  ----------
                                    1111001 (???)
```

Facendo il conto in base dieci, il risultato atteso è 67. Eseguendo il calcolo in base 2 invece il risultato conseguito è 1111001: il primo bit a uno indica che è un numero negativo, di valore assoluto pari a 57 in base 10, cioè -57. Tale errore è ovviamente dovuto ad un problema di rappresentazione: il risultato dell'operazione non può essere rappresentato con soli 7 bit.

ESERCIZIO 4