Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** e i file contenuti nello StartKit.
Nota: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire di tanto in tanto *"Rebuild All"*.

Uno stabilimento balneare vende dei prodotti ai propri clienti, a credito: un cliente può recarsi al bar, chiedere un gelato o un panino, e chiedere che la spesa venga addebitata sul suo conto.
Il manager dello stabilimento identifica in maniera univoca i clienti in base al loro **nome** (una stringa di al più 31 caratteri, senza spazi) e in base al numero del loro **ombrellone**.
Il manager memorizza i prodotti venduti ai clienti a credito in un file di testo, dove per ogni riga memorizza le seguenti informazioni: il **nome** del cliente; a seguire, seguito da uno spazio, il numero **dell'ombrellone**; a seguire separato da uno spazio, il **prodotto** acquistato a credito (una stringa di al più 31 caratteri, senza spazi); infine, ancora separato da uno spazio, **l'importo** (un float).
Il manager dello stabilimento memorizza, sempre sullo stesso file, anche gli importi pagati da un cliente, seguendo questa convenzione: ancora il nome del cliente all'inizio di una riga, poi il numero dell'ombrellone, a seguire un prodotto speciale con la stringa "PAGATO", e infine l'importo di quanto ha pagato il cliente. Riassumendo, nello stesso file si trovano quindi sia i debiti che i crediti di tutti i clienti.

Esercizio 1 – Strutture dati Cliente, Operazione, e funzioni di lett./scritt. (mod. element.h e bar.h/.c)

Si definiscano le strutture dati **`Cliente`** (per memorizzare i dati identificativi di un cliente, cioè il suo nome ed il numero del suo ombrellone) e **`Operazione`** (per memorizzare i dati relativi all'acquisto a credito o al pagamento, cioè: il cliente come struttura dati di tipo **`Cliente`**, il prodotto acquistato, e l'importo).

Si definisca la funzione:

```
Operazione * leggiTutte(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga da tale file tutte le informazioni relative alle operazioni effettuate e le restituisca tramite un array di strutture dati di tipo **`Operazione`** allocato dinamicamente (della dimensione minima necessaria). Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore. Qualora la funzione incontri dei problemi nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a **`NULL`**, e zero come valore della dimensione.

Si definisca la procedura:

```
void stampaOperazioni(Operazione * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Operazione`**, e la sua dimensione **`dim`**, stampi a video le informazioni riguardo le operazioni registrate nel file.

Il candidato definisca una procedura:

```
void ordina(Operazione * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Operazione`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine lessicografico crescente in base al nome del cliente; in caso di uguaglianza, in ordine lessicografico crescente in base al prodotto. A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "merge sort" visto a lezione.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

Esercizio 2 – Liste e Ordinamenti (moduli element.h/.c e bar.h/.c)

Si definisca la funzione:

```
list clienti(Operazione * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Operazione`**, e la dimensione **`dim`** di tale vettore, restituisca una lista di strutture dati di tipo **`Cliente`**, contenente i clienti che hanno effettuato operazioni memorizzate nel vettore. La lista non dovrà contenere ripetizioni: due clienti sono "uguali" se hanno lo stesso nome e lo stesso ombrellone.

Si definisca poi una funzione:

```
list ordinaLista(list clienti);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Cliente`**, restituisca una nuova lista di strutture dati di tipo **`Cliente`**, ma ordinata secondo il seguente criterio: in ordine alfabetico lessicografico in base al nome del cliente; in caso di stesso nome, in ordine crescente in base al numero di ombrellone.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Esercizio 3 – Calcolo del saldo dei clienti (modulo bar.h/bar.c)

Il manager dello stabilimento balneare vuole sapere, per ogni cliente, il saldo relativo a tale cliente, tenendo conto di tutti i prodotti acquistati, e di quanto eventualmente il cliente ha già pagato. A tal scopo, si definisca una procedura:

```
void saldi(Operazione * v, int dim);
```

che ricevuti come parametri un vettore di strutture dati di tipo **`Operazione`**, e la sua dimensione, stampi a video per ogni cliente il saldo positivo/negativo di quel cliente. La stampa a video non dovrà ripetere i clienti, e i clienti dovranno essere stampati in ordine in base al loro nome. Inoltre la funzione dovrà stampare anche il cliente con il saldo minore di tutti (che sarà negativo in caso il cliente abbia molti debiti).

Esercizio 4 Stampa dei saldi dei singoli clienti, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file i dati relativi alle operazioni, e stampi poi a video, per ogni cliente, il suo saldo relativo.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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

#define DIM_NOME 32
#define DIM_PRODOTTO 32

typedef struct {
	char nome[DIM_NOME];
	int ombrellone;
} Cliente;

typedef struct {
	Cliente cliente;
	char prodotto[DIM_PRODOTTO];
	float prezzo;
} Operazione;

typedef Cliente element;

int compareCliente(Cliente c1, Cliente c2);
int compareOperazione(Operazione o1, Operazione o2);

#endif
```

"element.c":

```
#include "element.h"

int compareCliente(Cliente c1, Cliente c2) {
	int result;

	result = strcmp(c1.nome, c2.nome);
	if (result==0)
		result = c1.ombrellone-c2.ombrellone;
	return result;
}

int compareOperazione(Operazione o1, Operazione o2) {
	int result;

	result = strcmp(o1.cliente.nome, o2.cliente.nome);
	if (result==0)
		result = strcmp(o1.prodotto, o2.prodotto);
	return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

"list.h"

```
#ifndef LIST_H
#define LIST_H

#include "element.h"
typedef struct list_element {
    element value;
    struct list_element  *next;
} item;

typedef  item* list;

/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);

void showlist(list l);
void freelist(list l);
int member(element el, list l);
list insord_p(element el, list l);
#endif
```

"list.c":

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

/* OPERAZIONI PRIMITIVE */
list  emptylist(void) {
    return NULL;
}

boolean  empty(list l) {
    return (l==NULL);
}

list cons(element e, list l) {
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
  return(t);
}

element head(list l) {
  if (empty(l)) exit(-2);
  else return (l->value);
}

list  tail(list l) {
  if (empty(l)) exit(-1);
  else return (l->next);
```

```
}

void showlist(list l) {
	element temp;
	if (!empty(l)) {
		temp = head(l);
		printf("%s %d\n", temp.nome, temp.ombrellone);
		showlist(tail(l));
		return;
	}
	else {
		printf("\n\n");
		return;
	}
}

int member(element el, list l) {
	int result = 0;
	while (!empty(l) && !result) {
		result = (compareCliente(el, head(l))==0);
		if (!result)
			l = tail(l);
	}
	return result;
}

void freelist(list l) {
	if (empty(l))
		return;
	else {
		freelist(tail(l));
		free(l);
	}
	return;
}

list insord_p(element el, list l) {
	list pprec, patt = l, paux;
	int trovato = 0;
	while (patt!=NULL && !trovato) {
		if (compareCliente(el, patt->value)<0)
			trovato = 1;
		else {
			pprec = patt;
			patt = patt->next;
		}
	}
	paux = (list) malloc(sizeof(item));
	paux->value = el;
	paux->next = patt;
	if (patt==l) return paux;
	else {
		pprec->next = paux;
		return l;
	}
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

"bar.h":

```
#ifndef _BAR_H
#define _BAR_H

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

// Es. 1
Operazione * leggiTutte(char * fileName, int * dim);
void stampaOperazioni(Operazione * v, int dim);
void ordina(Operazione * v, int dim);

// Es. 2
list clienti(Operazione * v, int dim);
list ordinaLista(list clienti);

// Es. 3
void saldi(Operazione * v, int dim);
#endif
```

"bar.c":

```
#include "element.h"
#include "bar.h"

// Es. 1
Operazione * leggiTutte(char * fileName, int * dim) {
	Operazione * result = NULL;
	FILE * fp;
	Operazione oTemp;
	int count;
	int i;

	*dim = 0;
	count = 0;
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		while (fscanf(fp,"%s%d%s%f", oTemp.cliente.nome,
&(oTemp.cliente.ombrellone), oTemp.prodotto, &(oTemp.prezzo)) == 4) {
			count++;
		}
		rewind(fp);
		if (count>0) {
			result = (Operazione*) malloc(sizeof(Operazione) * count);
			i = 0;
			while (fscanf(fp,"%s%d%s%f", oTemp.cliente.nome,
&(oTemp.cliente.ombrellone), oTemp.prodotto, &(oTemp.prezzo)) == 4 && i<count) {
				result[i] = oTemp;
				i++;
			}
			*dim = i;
		}
		fclose(fp);
	}
```

```
	else {
		printf("Problema nell'apertura del file %s\n", fileName);
	}
	return result;
}

void stampaOperazioni(Operazione * v, int dim) {
	int i;
	for (i=0; i<dim; i++) {
		printf("%s %d %s %f\n", v[i].cliente.nome, v[i].cliente.ombrellone,
v[i].prodotto, v[i].prezzo);
	}
 }

void  merge(Operazione v[], int i1, int i2, int fine, Operazione vout[]) {
	int i=i1, j=i2, k=i1;
	while ( i <= i2-1 &&  j <= fine ) {
		if (compareOperazione(v[i], v[j])<0)
			vout[k] = v[i++];
		else
			vout[k] = v[j++];
		k++;
	}
	while (i<=i2-1) { vout[k] = v[i++]; k++; }
	while (j<=fine) { vout[k] = v[j++]; k++; }
	for (i=i1; i<=fine; i++)
		v[i] = vout[i];
}

void mergeSort(Operazione v[], int first, int last, Operazione vout[]) {
	int mid;
	if ( first < last ) {
		mid = (last + first) / 2;
		mergeSort(v, first, mid,  vout);
		mergeSort(v, mid+1, last, vout);
		merge(v, first, mid+1, last, vout);
	}
}

void ordina(Operazione * v, int dim) {
	Operazione * temp;
	temp = (Operazione*) malloc(sizeof(Operazione)*dim);
	mergeSort(v, 0, dim-1, temp);
	free(temp);
}

// Es. 2
list clienti(Operazione * v, int dim) {
	list result;
	int i;
	result = emptylist();
	for (i=0; i<dim; i++) {
		if (!member(v[i].cliente, result))
			result = cons(v[i].cliente, result);
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

```
list ordinaLista(list clienti) {
	list result;
	Cliente temp;
	result = emptylist();
	while (!empty(clienti)) {
		temp = head(clienti);
		result = insord_p(temp, result);
		clienti = tail(clienti);
	}
	return result;
}

// Es. 3
void saldi(Operazione * v, int dim) {
	list c, cOrd, temp;
	float saldo;
	float saldoMin;
	int i;
	Cliente cliente;
	Cliente peggiore;
	int firstClient;

	c = clienti(v, dim);
	cOrd = ordinaLista(c);
	temp = cOrd;
	firstClient = 1;
	while (!empty(temp)) {
		cliente = head(temp);
		saldo = 0.0f;
		for (i=0; i<dim; i++) {
			if (compareCliente(cliente, v[i].cliente) == 0) {
				if (strcmp("PAGATO", v[i].prodotto) == 0)
					saldo = saldo + v[i].prezzo;
				else
					saldo = saldo - v[i].prezzo;
			}
		}
		printf("Saldo del cliente %s: %f\n", cliente.nome, saldo);
		temp = tail(temp);
		if (firstClient) {
			peggiore = cliente;
			saldoMin = saldo;
			firstClient = 0;
		}
		else {
			if (saldo <saldoMin) {
				saldoMin = saldo;
				peggiore = cliente;
			}
		}
	}
	printf("\n\nIl cliente con il saldo minimo e' %s, con saldo apri a %f\n",
peggiore.nome, saldoMin);

	freelist(c);
	freelist(cOrd);
```

```
        return;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>

#include "element.h"
#include "bar.h"

int main(int argc, char **argv)
{

	// Es. 1
	{
		Operazione *el;
		int dim;

		el = leggiTutte("bar.txt", &dim);
		stampaOperazioni(el, dim);

		printf("\n\n");
		ordina(el, dim);
		stampaOperazioni(el, dim);
		free(el);
	}

	// Es. 2
	{
		Operazione *el;
		int dim;
		list l, l2;

		el = leggiTutte("bar.txt", &dim);
		l = clienti(el, dim);
		showlist(l);
		l2 = ordinaLista(l);
		showlist(l2);

		free(el);
		freelist(l);
		freelist(l2);
	}

	// Es. 3
	{
		Operazione *el;
		int dim;

		el = leggiTutte("bar.txt", &dim);
		saldi(el, dim);

		free(el);
	}


	return 0;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 5A di Giovedì 13 Luglio 2017 – tempo a disposizione 2h

"bar.txt":

```
Federico 34 pizza 1.80
Federico 34 spritz 4.00
Paola 17 spritz 4.50
Daniela 99 piadina 4.50
Daniela 99 caffe 1.20
Andrea 56 piadina 4.50
Federico 34 ugo 4.00
Daniela 99 piadina 4.50
Daniela 99 pizza 4.50
Paola 17 gelato 3.50
Daniela 99 gelato 3.50
Paola 17 caffe 1.20
Luca 63 acqua 1.00
Paola 17 PAGATO 25.00
Daniela 99 cocco 1.00
Federico 34 piadina 4.50
Andrea 56 piadina 4.50
Andrea 56 piadina 4.50
Luca 63 piadina 4.50
Luca 63 PAGATO 10.00
Federico 34 PAGATO 10.00
Paola 17 racchettoni 19.00
Daniela 99 collanina 14.99
Daniela 99 PAGATO 20.00
Andrea 56 PAGATO 10.00
Luca 63 cocco 1.00
```