Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – 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"*.

Una piccola azienda produce e vende capsule e cialde di caffè per macchine automatiche. Riceve gli ordini dai clienti, preleva dal magazzino le capsule o le cialde richieste, e le consegna al cliente. A tal scopo memorizza gli ordini ricevuti in un file di testo, un ordine su ogni riga, con le seguenti informazioni: la **data** in cui è stato effettuato l'ordine (tre interi, nel formato aaaa/mm/gg); a seguire, separata da uno spazio, il **tipo** di prodotto ordinato (una stringa di al più 7 caratteri utili, senza spazi, che può assumere solo valori "cialda" o "capsula"); a seguire il **gusto** del prodotto ordinato (una stringa di al più 255 caratteri utili, senza spazi); ancora a seguire, separato da uno spazio, il **numero** di confezioni richieste (un intero); infine, ancora separato da uno spazio, l'identificatore unico del **cliente** (un intero positivo). Si veda, a titolo di esempio, il file di testo "ordini.txt" fornito nello StartKit: non è noto a priori quanti ordini siano memorizzati nel file.

In un secondo file di testo l'azienda memorizza i prezzi delle confezioni di cialde e di capsule: su ogni riga è presente il **tipo** (stringa di al più 7 caratteri, senza spazi, "cialda" o "capsula"), il **gusto** (stringa di al più 255 caratteri utili, senza spazi), e il **prezzo** (un float) per singola confezione. Si veda, a titolo di esempio, il file di testo "costi.txt" fornito nello StartKit.

Esercizio 1 – Strutture dati Data, Ordine, e funzioni di lett./scritt. (mod. element.h e ordini.h/.c)

Si definiscano delle opportune strutture dati **`Data`** e **`Ordine`** per memorizzare, rispettivamente, una data, ed un ordine di cialde/capsule per il caffè, memorizzando tutte le informazioni relative ad un ordine (data, tipo, gusto, numero di confezioni, e cliente).

Si definisca la funzione:

```
Ordine * leggiTutti(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente gli ordini, legga da tale file gli ordini ivi contenuti, e li restituisca tramite un array di strutture dati di tipo **`Ordine`** 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 nella lettura, o vi siano errori 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 noti che il file degli ordini contiene a volte degli errori: in particolare, vengono inseriti degli ordini con un tipo diverso da "capsula" o "cialda", oppure un numero di confezioni negativo, o ancora un identificatore del cliente negativo: in tali casi, l'ordine errato deve essere escluso dall'array restituito come risultato.

Si definisca la procedura:

```
void stampa(Ordine * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Ordine`**, e la sua dimensione **`dim`**, stampi a video i dati degli ordini.
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, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

Esercizio 2 –Ordinamento e lettura dei costi (moduli element.h/.c e ordini.h/.c)

Il candidato definisca una procedura:

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

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

Si definisca una struttura dati di nome **`Costo`**, che contenga tre campi: una stringa per determinare il tipo di caffè, una stringa per il gusto, ed il prezzo di una singola confezione.

Si definisca poi una funzione:

```
list leggiCosti(char * fileName);
```

che, ricevuto in ingresso il nome di un file contenente i prezzi dei vari tipi/gusti di caffè, restituisca in uscita una lista di strutture dati di tipo **`Costo`**, contenente l'elenco dei prezzi dei vari tipi di caffè. In caso di errore apertura del file, la funzione deve restituire una lista vuota.

Esercizio 3 – Calcolo fatture mensili (modulo ordini.h/ordini.c)

Per evitare troppa burocrazia, la piccola azienda emette fatture di pagamento solo ogni tanto. Gli ordini quindi si accumulano, anche relativamente a diversi mesi. L'azienda quindi necessita di stampare a video per ogni cliente e per ogni mese, l'ammontare di quanto dovuto da quel cliente. A tal scopo, si sviluppi una procedura:

```
void fatture(Ordine * v, int dim, list costi);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Ordine`** contenente gli ordini, la dimensione di tale vettore, e una lista di strutture dati di tipo **`Costo`**, stampi a video, per ogni cliente, per ogni mese, quanto dovuto dal cliente.

Si noti che la lista dei costi contiene i prezzi per confezione, e che ogni ordine prevede più confezioni: il costo del singolo ordine quindi sarà dato dal prodotto del costo della singola confezione per il numero di confezioni ordinate.

Si noti poi che ogni cliente ha fatto diversi ordini in uno stesso mese, ma la funzione deve stampare a video il totale dovuto da quel cliente in quel mese: il totale dovuto in un mese è dato dalla somma di quanto dovuto per i singoli ordini.

Infine, si noti che il file dei costi potrebbe non essere completo: ad esempio, il file fornito di esempio non contiene il prezzo delle "cialda adulto": in tal caso si assuma che il prezzo sia pari a zero. Ad esempio, usando i dati forniti di esempio nello StartKit, l'output atteso dovrà essere qualcosa simile a:

```
Cliente 123, Mese: 2017/12, Totale dovuto:                  118.80
Cliente 123, Mese: 2018/1, Totale dovuto:                   163.00
Cliente 234, Mese: 2017/12, Totale dovuto:                  0.00
Cliente 234, Mese: 2018/1, Totale dovuto:                   516.00
```

Esercizio 4 - Stampa dei risultati, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file tutte le informazioni e stampi quanto dovuto da parte di ogni cliente per ogni mese.

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, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_TIPO 8
#define DIM_GUSTO 255

typedef struct {
	int anno;
	int mese;
	int giorno;
} Data;
typedef struct {
	Data data;
	char tipo[DIM_TIPO];
	char gusto[DIM_GUSTO];
	int num;
	int cliente;
} Ordine;
typedef struct {
	char tipo[DIM_TIPO];
	char gusto[DIM_GUSTO];
	float prezzo;
} Costo;
typedef Costo element;

int compare(Ordine o1, Ordine o2);
#endif
```

"element.c":

```
#include "element.h"
#include <string.h>

int compareData(Data d1, Data d2) {
	int result;
	result = d1.anno-d2.anno;
	if (result == 0)
		result = d1.mese-d2.mese;
	if (result == 0)
		result = d1.giorno-d2.giorno;
	return result;
}

int compare(Ordine o1, Ordine o2) {
	int result;

	result = o1.cliente-o2.cliente;
	if (result == 0)
		result = compareData(o1.data, o2.data);
	if (result == 0)
		result = strcmp(o1.tipo, o2.tipo);
	if (result == 0)
		result = strcmp(o1.gusto, o2.gusto);
	return result;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – 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;

typedef  int boolean;

/* 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);
#endif
```

"list.c":

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

/* OPERAZIONI PRIMITIVE */
list  emptylist(void) {          /* costruttore lista vuota */
    return NULL;
}
boolean  empty(list l) {   /* verifica se lista vuota */
    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) { /* selettore testa lista */
  if (empty(l)) exit(-2);
  else return (l->value);
}
list  tail(list l) {          /* selettore coda lista */
  if (empty(l)) exit(-1);
  else return (l->next);
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

```
void showlist(list l) {
	element temp;
	if (!empty(l)) {
		temp = head(l);
		printf("%s %s %6.2f\n",
			temp.tipo, temp.gusto, temp.prezzo);
		showlist(tail(l));
		return;
	}
	else {
		printf("\n\n");
		return;
	}
}

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

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

"ordini.h":

```
#ifndef _ORDINI_H
#define _ORDINI_H

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

// Es. 1
Ordine * leggiTutti(char * fileName, int * dim);
void stampa(Ordine * v, int dim);
// Es. 2
void ordina(Ordine * v, int dim);
list leggiCosti(char * fileName);
// Es. 3
void fatture(Ordine * v, int dim, list costi);

#endif
```

"ordini.c":

```
#include "ordini.h"

// Es. 1
int corretto(Ordine o) {
	if (
		(strcmp("cialda", o.tipo) && strcmp("capsula", o.tipo))
		|| o.num <= 0
		|| o.cliente <= 0
		)
		return 0;
	else
		return 1;

}

Ordine * leggiTutti(char * fileName, int * dim) {
	FILE * fp;
	Ordine * result;
	Ordine temp;
	int i;

	*dim = 0;
	i = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file %s\n", fileName);
		result = NULL;
	}
	else {
		while (fscanf(fp, "%d/%d/%d%s%s%d%d",
			&(temp.data.anno), &(temp.data.mese), &(temp.data.giorno),
			temp.tipo, temp.gusto, &(temp.num), &(temp.cliente)) == 7) {
				if (corretto(temp)) {
```

```
                        *dim = *dim + 1;
                    }
                }
                rewind(fp);
                result = (Ordine *) malloc(sizeof(Ordine) * *dim);
                while (fscanf(fp, "%d/%d/%d%s%s%d%d",
                        &(temp.data.anno), &(temp.data.mese), &(temp.data.giorno),
                        temp.tipo, temp.gusto, &(temp.num), &(temp.cliente)) == 7) {
                        if (corretto(temp)) {
                                result[i] = temp;
                                i++;
                        }
                }
                fclose(fp);
        }
        return result;
}

void stampa(Ordine * v, int dim) {
        int i;

        for (i=0; i<dim; i++) {
                printf("%d/%d/%d %s %s %d %d\n",
                        v[i].data.anno, v[i].data.mese, v[i].data.giorno,
                        v[i].tipo, v[i].gusto, v[i].num, v[i].cliente);
        }
        return;
}


// Es. 2

void  merge(Ordine v[], int i1, int i2, int fine, Ordine vout[]) {
        int i=i1, j=i2, k=i1;
        while ( i <= i2-1 &&  j <= fine ) {
                if (compare(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(Ordine v[], int first, int last, Ordine 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);
        }
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

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


list leggiCosti(char * fileName) {
	list result = emptylist();
	FILE * fp;
	Costo temp;

	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		while (fscanf(fp, "%s%s%f", temp.tipo, temp.gusto, &(temp.prezzo)) == 3) {
			result = cons(temp, result);
		}
		fclose(fp);
	}
	else {
		printf("Errore nell'apertura del file %s\n", fileName);
	}
	return result;
}


// Es. 3
float costoOrdine(Ordine o, list costi) {
	float result;
	float prezzo;
	int trovato = 0;

	result = 0;
	while (!empty(costi) && !trovato) {
		if (strcmp(o.tipo, head(costi).tipo) == 0 &&
			strcmp(o.gusto, head(costi).gusto) == 0 )
			trovato = 1;
		else
			costi = tail(costi);
	}
	if (trovato)
		result = o.num * head(costi).prezzo;
	else
		result = 0.0F;
	return result;
}

void fatture(Ordine * v, int dim, list costi) {
	Data tData;
	int tCliente;
	float totale;
	int i;

	ordina(v, dim);

	if (dim > 0) {
		tData = v[0].data;
```

```
            tCliente = v[0].cliente;
            totale = costoOrdine(v[0], costi);
            for (i=1; i<dim; i++) {
                  if (v[i].data.anno != tData.anno || v[i].data.mese != tData.mese ||
v[i].cliente != tCliente) {
                        printf("Cliente %d, Mese: %d/%d, Totale dovuto: %10.2f\n",
tCliente, tData.anno, tData.mese, totale);
                        tData = v[i].data;
                        tCliente = v[i].cliente;
                        totale = costoOrdine(v[i], costi);
                  }
                  else {
                        totale = totale + costoOrdine(v[i], costi);
                  }
            }
            // stampol'ultimo cliente/mese
            printf("Cliente %d, Mese: %d/%d, Totale dovuto: %10.2f\n", tCliente,
tData.anno, tData.mese, totale);
      }
      return;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

"main.c":

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


#include "element.h"
#include "ordini.h"



int main() {
	// Es. 1
	{
		Ordine * ord;
		int dim;
		ord = leggiTutti("ordini.txt", &dim);
		stampa(ord, dim);
		free(ord);
	}
	// Es. 2
	{
		Ordine * ord;
		int dim;
		list costi;
		ord = leggiTutti("ordini.txt", &dim);
		ordina(ord, dim);
		printf("\n\n\n");
		stampa(ord, dim);
		costi=leggiCosti("costi.txt");
		showlist(costi);
		freelist(costi);
		free(ord);
	}

	// Es. 3
	{
		Ordine * ord;
		int dim;
		list costi;
		ord = leggiTutti("ordini.txt", &dim);
		costi=leggiCosti("costi.txt");
		printf("\n\n\n");
		fatture(ord, dim, costi);
		freelist(costi);
		free(ord);
	}

	return 0;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 2A di Giovedì 25 Gennaio 2018 – tempo a disposizione 2h

"ordini.txt":

```
2018/01/22 capsula vaniglia 10 123
2018/01/22 cialda cappuccino 10 123
2018/01/24 capsula dark_sensation 20 234
2018/01/22 capsula nero_arabica 15 123
2018/01/23 capsula decaffeinato -5 123
2018/01/23 capsula decaffeinato 5 -999
2018/01/23 pillola decaffeinato 5 123
2018/01/24 cialda velvet 20 234
2017/12/30 cialda dark_sensation 12 123
2017/12/24 cialda adulto 20 234
```

"costi.txt":

```
capsula vaniglia 2.30
cialda cappuccino 2.60
capsula dark_sensation 9.90
capsula nero_arabica 7.60
cialda velvet 15.90
cialda dark_sensation 9.90
```