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

Prova d'Esame 5A di Giovedì 12 Luglio 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 libreria tradizionale ha deciso di aprire un servizio di acquisto libri on-line. A tal scopo, gli ordini ricevuti tramite un'interfaccia web sono memorizzati su un file di testo, che poi deve essere elaborato per permettere agli impiegati della libreria di soddisfare gli ordini ricevuti. Un ordine è tipicamente composto di uno o più libri: nel file di testo compaiono i singoli libri, uno e un solo libro su ogni riga. Quindi più righe del file possono essere relative ad uno stesso ordine.
Nel file di testo, su ogni riga, sono contenute le seguenti informazioni: innanzitutto un codice unico identificativo dell'ordine (un intero); a seguire, separato da uno spazio, un codice unico identificativo dell'utente (un intero); poi, separato da uno spazio, il titolo del libro (una stringa di al più 511 caratteri utili, SENZA spazi); infine, ancora separato da uno spazio, il prezzo del libro (un float).
Si veda, a titolo di esempio, il file di testo "**`libri.txt`**" fornito nello StartKit: non è noto a priori quanti libri e quanti ordini siano memorizzati nei file.

Esercizio 1 – Struttura dati Libro, e funzioni di lett./scritt. (mod. element.h e libri.h/.c)

Si definisca un'opportuna struttura dati **`Libro`** per memorizzare i dati relativi ad un libro ordinato. In particolare, la struttura dovrà comprendere un campo per l'identificatore dell'ordine, un campo per l'identificatore dell'utente, un campo per il titolo, ed un campo per il prezzo del libro.

Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file contenente l'elenco dei libri ordinati, legga da tale file i dati ivi contenuti, e li restituisca tramite una lista di strutture dati di tipo **`Libro`**. Qualora s'incontrino dei problemi nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video e restituire una lista vuota.

Si definisca la procedura RICORSIVA:

```
void stampaLibri(list elencoLibri);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Libro`**, stampi a video l'elenco dei libri ordinati.

Purtroppo, l'applicazione web che raccoglie gli ordini ogni tanto presenta un bug fastidioso, ed uno stesso libro per uno stesso ordine può comparire più volte nel file. Si rende quindi necessario eliminare tali righe ripetute, lasciando soltanto una e una sola apparizione di un certo libro, per un certo ordine, per un certo utente. A tal scopo si definisca infine la funzione:

```
list filtra(list elencoLibri);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Libro`**, restituisca una nuova lista dove i libri ripetuti siano stati eliminati, e tali libri compaiano una sola volta. Una struttura dati **`Libro`** è da considerarsi ripetuta se è presente con lo stesso id di ordine, lo stesso id di utente, e lo stesso titolo libro.

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 de-allocare la memoria, se necessario.

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

Prova d'Esame 5A di Giovedì 12 Luglio 2018 – tempo a disposizione 2h

Esercizio 2 – Struttura dati Ordine, e Ordinamento (modulo libri.h/.c)

Si definisca nel file "libri.h" una struttura dati **`Ordine`**, atta a memorizzare le informazioni relative ad un ordine: l'identificatore dell'ordine, l'identificatore dell'utente, e la lista di libri relativi a quell'ordine e a quell'utente.

Il candidato definisca una procedura:

```
Ordine estrai(list elencoLibri, int idOrdine);
```

che, ricevuto in ingresso una lista di strutture dati di tipo Libro, e un identificatore di ordine, restituisca una struttura dati **`Ordine`** dove siano stati indicati correttamente gli identificatori di ordine e di cliente, e che contenga una NUOVA lista contenete solo i libri relativi all'ordine specificato.

Il candidato definisca una procedura:

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

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Ordine`**, e la dimensione **`dim`** di tale vettore, ordini il vettore secondo il seguente criterio: in ordine crescente rispetto all'identificatore dell'utente; a parità di utente (un utente infatti può aver fatto diversi ordini), si ordini in senso decrescente in base al numero di libri ordinati nell'**`Ordine`** specifico (cioè, la lunghezza della lista nella struttura dati **`Ordine`**). A tal scopo, si utilizzi uno qualunque tra gli algoritmi di ordinamento visti a lezione.
SUGGERIMENTO: si definisca una funzione **`int compare(Ordine o1, Ordine o2)`** nel modulo "libri.h/libri.c".

Esercizio 3 – Creazione del vettore degli Ordini (modulo libri.h/libri.c)

Ai fini della fatturazione, la libreria deve ottenere dal file di testo l'elenco degli ordini. A tal scopo, si sviluppi la funzione:

```
Ordine * elenco(list elencoLibri, int * dim);
```

che, ricevuti in ingresso una lista di strutture dati di tipo **`Libro`**, contenente tutti i libri memorizzati nel file, restituisca un array allocato dinamicamente, della dimensione minima necessaria, di strutture dati di tipo **`Ordine`**. Tramite il parametro **`dim`**, passato per riferimento, la funzione restituisca la dimensione del vettore allocato. Il vettore restituito dovrà contenere gli ordini, estratti dalla lista dei libri, tramite l'utilizzo della funzione **`estrai(…)`** definita al punto precedente.

Esercizio 4 – Stampa degli ordini, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che stampi a video l'elenco degli ordini, ordinati secondo i criteri specificati al punto 2, avendo cura di eliminare gli eventuali libri ripetuti (l'eliminazione dei ripetuti dovrà avvenire prima di invocare la funzione **`s`**). 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 5A di Giovedì 12 Luglio 2018 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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

#define DIM 512

typedef struct {
	int idOrdine;
	int idUtente;
	char titolo[DIM];
	float prezzo;
} Libro;

typedef Libro element;

int equals (Libro l1, Libro l2);

#endif
```

"element.c":

```
#include "element.h"


int equals (Libro l1, Libro l2) {
	return (l1.idOrdine == l2.idOrdine)
		&& (l1.idUtente == l2.idUtente)
		&& strcmp(l1.titolo, l2.titolo)==0;
}
```

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

Prova d'Esame 5A di Giovedì 12 Luglio 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);
int member(element el, list l);
int length(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)            /* 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);
}

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

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

int length(list l) {
       if (empty(l))
              return 0;
       else
              return 1 + length(tail(l));
}

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 5A di Giovedì 12 Luglio 2018 – tempo a disposizione 2h

"libri.h":

```
#ifndef _LIBRI_H
#define _LIBRI_H

#include "element.h"
#include "list.h"

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

typedef struct {
	int idOrdine;
	int idUtente;
	list elencoLibri;
} Ordine;



// Es. 1
list leggi(char * fileName);
void stampaLibri(list elencoLibri);
list filtra(list elencoLibri);

// Es. 2
Ordine estrai(list elencoLibri, int idOrdine);
void ordina(Ordine * v, int dim);

// Es. 3
Ordine * elenco(list elencoLibri, int * dim);

#endif
```

"libri.c":

```
#include "libri.h"


// Es. 1
list leggi(char * fileName) {
	FILE * fp;
	list result;
	Libro temp;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp!=NULL) {
		while (fscanf(fp, "%d %d %s %f", &(temp.idOrdine), &(temp.idUtente),
temp.titolo, &(temp.prezzo)) == 4) {
			result = cons(temp, result);
		}
	}
	else {
		printf("Errore durante l'apertura del file %s\n", fileName);
	}
	return result;
```

```
}

void stampaLibri(list elencoLibri) {
	element temp;
	if (!empty(elencoLibri)) {
		temp = head(elencoLibri);
		printf("%d %d %s %6.2f\n",
			temp.idOrdine, temp.idUtente, temp.titolo, temp.prezzo);
		stampaLibri(tail(elencoLibri));
		return;
	}
	else {
		printf("\n\n");
		return;
	}
}

list filtra(list elencoLibri) {
	list result;
	Libro temp;

	result = emptylist();
	while (!empty(elencoLibri)) {
		temp = head(elencoLibri);
		if (!member(temp, result)) {
			result = cons(temp, result);
		}
		elencoLibri = tail(elencoLibri);
	}
	return result;
}


// Es. 2
Ordine estrai(list elencoLibri, int idOrdine) {
	Ordine result;
	Libro temp;

	result.idOrdine = idOrdine;
	result.elencoLibri = emptylist();
	while (!empty(elencoLibri)) {
		temp = head(elencoLibri);
		if (temp.idOrdine == idOrdine) {
			result.elencoLibri = cons(temp, result.elencoLibri);
			result.idUtente = temp.idUtente;
		}
		elencoLibri = tail(elencoLibri);
	}
	return result;
}

int compare(Ordine o1, Ordine o2) {
	int result;
	result = o1.idUtente - o2.idUtente;
	if (result == 0) {
		result = length(o2.elencoLibri)-length(o1.elencoLibri);
	}
```

```
}

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);
       }
}

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


// Es. 3

int contains(int idOrdine, list elenco) {
       if (empty(elenco))
              return 0;
       else
              if (head(elenco).idOrdine == idOrdine)
                     return 1;
              else
                     return contains(idOrdine, tail(elenco));
}

Ordine * elenco(list elencoLibri, int * dim) {
       Ordine * result;
       int count;
       int idOrdine;
       int i;
       list temp;

        *dim = 0;
        count = 0;
        result = NULL;
        // conta degli ordini...
        temp = elencoLibri;
        while (!empty(temp)) {
```

```
            if (!contains(head(temp).idOrdine, tail(temp)))
                  count++;
            temp = tail(temp);
      }

      if (count>0) {
            temp = elencoLibri;
            i = 0;
            result = (Ordine *) malloc(sizeof(Ordine) * count);
            while (!empty(temp)) {
                  if (!contains(head(temp).idOrdine, tail(temp)))
                        result[i++] = estrai(elencoLibri, head(temp).idOrdine);
                  temp = tail(temp);
            }
            *dim = count;
      }
      return result;
}
```

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

Prova d'Esame 5A di Giovedì 12 Luglio 2018 – tempo a disposizione 2h

"main.c":

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

int main() {
	{ // Es. 1
		list elencoLibri;
		list filtrato;
		elencoLibri = leggi("libri.txt");
		stampaLibri(elencoLibri);
		filtrato = filtra(elencoLibri);
		stampaLibri(filtrato);
		freelist(elencoLibri);
		freelist(filtrato);
	}
	{ // Es. 2
		list elencoLibri;
		list filtrato;
		Ordine unOrdine;
		elencoLibri = leggi("libri.txt");
		filtrato = filtra(elencoLibri);
		unOrdine = estrai(filtrato, 123);
		printf("Ordine: %d\nUtente: %d\n", unOrdine.idOrdine, unOrdine.idUtente);
		stampaLibri(unOrdine.elencoLibri);
		freelist(elencoLibri);
		freelist(filtrato);
		freelist(unOrdine.elencoLibri);
	}
	{ // Es. 3
		list elencoLibri;
		list filtrato;
		Ordine * ordini;
		int dim;
		int i;
		elencoLibri = leggi("libri.txt");
		filtrato = filtra(elencoLibri);
		ordini = elenco(filtrato, &dim);
		if (dim>0) {
			for (i=0; i<dim; i++) {
				printf("IdOrdine: %d\nIdUtente: %d\nLibri:\n",
ordini[i].idOrdine, ordini[i].idUtente);
				stampaLibri(ordini[i].elencoLibri);
				printf ("\n\n");
			}
		}

		if (dim>0) {
			for (i=0; i<dim; i++) {
				freelist(ordini[i].elencoLibri);
			}
		}
		free(ordini);
		freelist(elencoLibri);
```

```
            freelist(filtrato);
        }

        return 0;
}
```

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

Prova d'Esame 5A di Giovedì 12 Luglio 2018 – tempo a disposizione 2h

"libri.txt":

```
123 45 Recommender_Systems_The_Textbook 51.42
123 45 Recommender_Systems_The_Textbook 51.42
123 45 Recommender_Systems_The_Textbook 51.42
123 45 Berta_Isla 18.70
123 45 Avventure_prodigi_e_meraviglie_Storia_di_re_Artu_e_dei_suoi_cavalieri 9.18
567 45 Business_Process_Management_Concepts_Languages_Architectures 55.52
123 45 The_Lean_Startup 10.84
890 87 Deep_Learning 54.90
```