Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – 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"*.

Un negozio di giocattoli registra l'elenco dei giocattoli che ha a disposizione in un file di testo. In particolare dedica una riga del file ad ogni giocattolo diverso, memorizzando le seguenti informazioni: un **identificatore** unico del giocattolo (un intero); separato da uno spazio, il **nome** del giocattolo (una stringa di al più 1023 caratteri utili, senza spazi); a seguire, separato da uno spazio, il **prezzo** unitario del giocattolo (un float); ancora separato da uno spazio, il **numero** di confezioni presenti in negozio (un intero). Ogni riga (compresa l'ultima) è terminata da un carattere 'newline'. Non è noto a priori quanti elementi siano memorizzati nel file.
In un secondo file vengono memorizzate le prenotazioni da parte dei clienti. In particolare in ogni riga viene scritto il **nome** del giocattolo di interesse, e a seguire separato da uno spazio il nome del **cliente** (ancora 1023 caratteri utili, senza spazi). Per problemi software, può accadere che la stessa prenotazione (stesso gioco, stesso cliente) compaia più volte nel file: tali ripetizioni sono da considerarsi un errore.
Si vedano, a titolo di esempio, i file di testo "negozio.txt" e "prenotazioni.txt" forniti nello StartKit.

Esercizio 1 – Strutture dati Gioco, Prenotazione, e funzioni di lett./scritt. (mod. element.h e negozio.h/.c)

Si definisca un'opportuna struttura dati **`Gioco`**, al fine di rappresentare un giocattolo definito come identificatore, nome, prezzo e numero di scatole disponibili. Si definisca un'opportuna struttura dati **`Prenotazione`**, al fine di rappresentare le informazioni relative ad una prenotazione (nome del giocattolo, cliente).
Si definisca la funzione:

```
Gioco * leggiGiocattoli(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente le informazioni relative ai giocattoli disponibili in negozio, legga da tale file tali informazioni, e le restituisca tramite un array allocato dinamicamente (di dimensione minima possibile) di strutture dati di tipo **`Gioco`**. Tramite il parametro **`dim`** passato per riferimento la funzione deve restituire la dimensione dell'array allocato dinamicamente. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve restituire un puntatore a NULL e come valore della dimensione il valore zero.
Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file di testo contenente le prenotazioni, restituisca una lista di strutture dati di tipo **`Prenotazione`**, contenente le informazioni presenti nel file il cui nome è passato come parametro. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve restituire una lista vuota.
Si definisca la procedura:

```
void stampaGiocattoli(Gioco * v, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Gioco`**, e la sua dimensione **`dim`**, stampi a video i giocattoli disponibili in negozio.
Si definisca la procedura:

```
void stampaPrenotazioni(list p);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Prenotazione`**, stampi a video le prenotazioni.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – tempo a disposizione 2h

Esercizio 2 – Ordinamento e gestione dei ripetuti(moduli element.h/.c e negozio.h/.c)

Il candidato definisca una procedura:

```
void ordinaGiochi(Gioco * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **Gioco** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine crescente in base al numero di scatole disponibili; a parità di numero di scatole disponibili in negozio, in ordine lessicografico in base al nome dei giocattoli. A tal fine, il candidato utilizzi l'algoritmo "bubble sort" visto a lezione.

Il candidato definisca una funzione:

```
list ripetuti(list p);
```

che, ricevuti in ingresso una lista di strutture dati di tipo **Prenotazione**, crei una nuova lista contenente i dati in ingresso, ma senza ripetizioni. Una prenotazione è identica ad un'altra (cioè è una ripetizione) se riguardano lo stesso nome del giocattolo, e lo stesso cliente.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Esercizio 3 – Calcolo richieste e giocattoli da comprare (modulo negozio.h/negozio.c)

Si vuole realizzare una funzionalità che permetta al negozio di determinare se è in grado di soddisfare tutte le prenotazioni, e che fornisca in uscita quali sono i giocattoli da comprare eventualmente per coprire tutte le prenotazioni. Si definisca dapprima una funzione:

```
int calcolaRichiesta(char * nome, list p);
```

che ricevuto in ingresso il nome di un giocattolo e la lista delle prenotazioni, restituisce il numero di scatole totali richieste per il gioco specificato.

Il candidato definisca la funzione:

```
Gioco * daComprare(Gioco * v, int dim, list p, int * dimResult);
```

che, ricevuti come parametri un vettore di strutture dati di tipo **Gioco** e la sua dimensione, e una lista di strutture dati di tipo **Prenotazione**, restituisca un nuovo vettore allocato dinamicamente (non necessariamente della dimensione minima possibile), contenente l'elenco dei giochi che sono prenotati, ma non sono disponibili in maniera sufficiente in negozio. In particolare, le strutture dati restituite come risultato dovranno indicare tramite il campo relativo alle disponibilità il numero di scatole da ordinare affinché il negozio possa soddisfare gli ordini. Tramite la variabile dimResult, la funzione dovrà restituire la dimensione del vettore restituito come risultato.

Esercizio 4 – Stampa dei giocattoli da acquistare, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **main(…)** un programma che, usando le informazioni fornite tramite i file di esempio fornito nello StartKit e le funzioni definite agli esercizi precedenti, stampi a video l'elenco dei giocattoli da acquistare affinché il negozio possa soddisfare le prenotazioni ricevute. A tal riguardo, si abbia cura di eliminare le prenotazioni ripetute.

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, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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


typedef struct {
	int id;
	char nome[1024];
	float prezzo;
	int num;
} Gioco;

typedef struct {
	char nome[1024];
	char cliente[1024];
} Prenotazione;


typedef Prenotazione element;

int compare(Gioco g1, Gioco g2);
int equals(Prenotazione p1, Prenotazione p2);

#endif
```

"element.c":

```
#include "element.h"


int compare(Gioco g1, Gioco g2) {
	int result;

	result = g1.num - g2.num;
	if (result == 0)
		result = strcmp(g1.nome, g2.nome);
	return result;
}

int equals(Prenotazione p1, Prenotazione p2) {
	return (
		strcmp(p1.nome, p2.nome) == 0
		&&
		strcmp(p1.cliente, p2.cliente) == 0
	);
}
```

Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – 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);

#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);
}


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;
}



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

Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – tempo a disposizione 2h

"negozio.h":

```
#ifndef _NEGOZIO_H
#define _NEGOZIO_H

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

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

Gioco * leggiGiocattoli(char * fileName, int * dim);
list leggiPrenotazione(char* fileName);
void stampaGiocattoli(Gioco * v, int dim);
void stampaPrenotazioni(list p);

void ordinaGiochi(Gioco * v, int dim);
list ripetuti(list p);

int calcolaRichiesta(char * nome, list p);
Gioco * daComprare(Gioco * v, int dim, list p, int * dimResult);

#endif
```

"calcio.c":

```
#include "negozio.h"


Gioco * leggiGiocattoli(char * fileName, int * dim) {
	FILE * fp;
	Gioco * result = NULL;
	int cont = 0;
	Gioco temp;
	int i = 0;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		return result;
	}
	else {
		while (fscanf(fp, "%d %s %f  %d", &(temp.id), temp.nome, &(temp.prezzo),
&(temp.num)) == 4)
			cont++;
		if (cont>0) {
			rewind(fp);
			result = (Gioco *) malloc(sizeof(Gioco) * cont);
			while (fscanf(fp, "%d %s %f  %d", &(temp.id), temp.nome,
&(temp.prezzo), &(temp.num)) == 4) {
				result[i] = temp;
				i++;
			}
			*dim = i;
		}
```

```
		fclose(fp);
	}
	return result;
}

list leggiPrenotazione(char* fileName) {
	FILE * fp;
	list result = emptylist();
	Prenotazione temp;

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		return result;
	}
	else {
		while (fscanf(fp, "%s %s", temp.nome, temp.cliente) == 2)
			result = cons(temp, result);
		fclose(fp);
		return result;
	}
}

void stampaGiocattoli(Gioco * v, int dim) {
	int i;
	for (i=0; i<dim; i++)
		printf("%d %s %f %d\n", v[i].id, v[i].nome, v[i].prezzo, v[i].num);
	printf("\n\n");
}

void stampaPrenotazioni(list p) {
	if (empty(p)) {
		printf("\n\n");
		return;
	}
	else {
		printf("%s %s\n", head(p).nome, head(p).cliente);
		return stampaPrenotazioni(tail(p));
	}
}

void scambia(Gioco *a, Gioco *b) {
	Gioco tmp = *a;
	*a = *b;
	*b = tmp;
}

void bubbleSort(Gioco v[], int n){
	int i, ordinato = 0;
	while (n>1 && !ordinato) {
		ordinato = 1;
		for (i=0; i<n-1; i++)
			if (compare(v[i],v[i+1]) > 0) {
				scambia(&v[i],&v[i+1]);
				ordinato = 0;
			}
		n--;
	}
```

```
}
void ordinaGiochi(Gioco * v, int dim) {
	bubbleSort(v, dim);
}

list ripetuti(list p) {
	list result = emptylist();
	Prenotazione temp;

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

int calcolaRichiesta(char * nome, list p) {
	int cont = 0;
	Prenotazione temp;

	while (!empty(p)) {
		temp = head(p);
		if (strcmp(nome, temp.nome) == 0)
			cont++;
		p = tail(p);
	}
	return cont;
}

Gioco * daComprare(Gioco * v, int dim, list p, int * dimResult) {
	Gioco * result;
	int i, k;
	int richiesta;

	*dimResult = 0;
	k = 0;
	result = (Gioco *) malloc(sizeof(Gioco) * dim);
	for (i=0; i<dim; i++) {
		richiesta = calcolaRichiesta(v[i].nome, p);
		if (v[i].num < richiesta) {
			result[k] = v[i];
			result[k].num = richiesta – v[i].num;
			k++;
		}
	}
	*dimResult = k;
	return result;
}
```

Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>

#include "element.h"
#include "negozio.h"

int main() {
	{ // Es. 1
		Gioco * magazzino;
		int dim;
		list p;
		magazzino = leggiGiocattoli("negozio.txt", &dim);
		stampaGiocattoli(magazzino, dim);
		p = leggiPrenotazione("prenotazioni.txt");
		stampaPrenotazioni(p);
		free(magazzino);
		freelist(p);
	}
	{ // Es. 2
		Gioco * magazzino;
		int dim;
		list p;
		list p1;
		magazzino = leggiGiocattoli("negozio.txt", &dim);
		ordinaGiochi(magazzino, dim);
		stampaGiocattoli(magazzino, dim);
		p = leggiPrenotazione("prenotazioni.txt");
		p1 = ripetuti(p);
		stampaPrenotazioni(p1);
		free(magazzino);
		freelist(p);
		freelist(p1);
	}
	{ // Es. 3 && 4
		Gioco * magazzino;
		Gioco * daOrdinare;
		int dim;
		int dimOrd;
		list p;
		list p1;
		magazzino = leggiGiocattoli("negozio.txt", &dim);
		ordinaGiochi(magazzino, dim);
		p = leggiPrenotazione("prenotazioni.txt");
		p1 = ripetuti(p);
		daOrdinare = daComprare(magazzino, dim, p1, &dimOrd);
		printf("DA ORDINARE:\n");
		stampaGiocattoli(daOrdinare, dimOrd);
		free(magazzino);
		free(daOrdinare);
		freelist(p);
		freelist(p1);
	}
	return 0;
}
```

Fondamenti di Informatica T-1, 2015/2016 – Modulo 2

Prova d'Esame 3A di Venerdì 12 Febbraio 2016 – tempo a disposizione 2h

"negozio.txt":

```
12357 LegoStarWarsMillenniumFalcum 159.99 3
10897 LegoStarWarsX-WingFighter 97.54 2
41445 BarbieMagiaDiNatale 39.80 5
70500 MinionsPeluche 20.90 6
23001 Monopoly 31.52 11
```

"prenotazioni.txt":

```
LegoStarWarsMillenniumFalcum Federico
LegoStarWarsX-WingFighter Francesco
LegoStarWarsX-WingFighter Chiara
LegoStarWarsMillenniumFalcum Federico
LegoStarWarsMillenniumFalcum Federico
LegoStarWarsMillenniumFalcum Federico
LegoStarWarsX-WingFighter Giovanni
LegoStarWarsX-WingFighter Filippo
LegoStarWarsX-WingFighter Matteo
LegoStarWarsMillenniumFalcum Giovanni
LegoStarWarsMillenniumFalcum Filippo
LegoStarWarsMillenniumFalcum Matteo
LegoStarWarsMillenniumFalcum Alessio
LegoStarWarsX-WingFighter Giovanni
LegoStarWarsX-WingFighter Giovanni
LegoStarWarsX-WingFighter Giovanni
```