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

Prova d'Esame 3A di Venerdì 9 Febbraio 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 grande banca italiana vuole fare pubblicità dei propri prodotti finanziari ai propri clienti, tramite telefonate mirate. A tal scopo, ha memorizzato su un file di testo i dati relativi ai clienti che potrebbero essere interessati ai prodotti. In particolare, su un file di testo, memorizza i dati di un cliente su ogni riga: innanzitutto un **identificativo** unico del cliente (un intero); a seguire, separato da uno spazio, il **nome** del cliente (una stringa di al più 255 caratteri utili, contenente spazi); separato da un carattere ';' il **cognome** del cliente (una stringa di al più 255 caratteri utili, contenente spazi); dopo un carattere ';' il numero di **telefono** (una stringa di al più 15, caratteri senza spazi); e infine, dopo uno spazio, il **saldo** del c/c del cliente (un float).

In un secondo file la banca memorizza i prodotti finanziari, con le seguenti informazioni per ogni prodotto: un **identificatore** unico del prodotto (un intero), uno spazio di separazione, il **nome** del prodotto (una stringa di al più 15 caratteri utili, senza spazi), uno spazio di separazione e il **costo** del prodotto (un float). I prodotti sono stati scritti nel file da diversi impiegati del settore marketing, e così accade a volte che su una stessa riga compaiano più prodotti, separati tra loro da uno spazio; accade anche che vi siano nel file righe semplicemente vuote.

Si vedano, a titolo di esempio, i file di testo "clienti.txt" e "prodotti.txt" forniti nello StartKit: non è noto a priori quanti dati siano memorizzati nei file.

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

Si definiscano delle opportune strutture dati **`Cliente`** e **`Prodotto`** per memorizzare, rispettivamente, i dati relativi ai clienti ed ai prodotti.

Si definisca la funzione:

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

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

Si definisca la funzione:

```
Prodotto * leggiProd(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente l'elenco dei prodotti, legga da tale file i dati ivi contenuti, e li restituisca tramite un array di strutture dati di tipo **`Prodotto`**, allocato dinamicamente e della dimensione minima necessaria. Tramite il parametro dim, passato per riferimento, la funzione dovrà restituire la dimensione dell'array allocato dinamicamente. Qualora la funzione incontri dei problemi nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video e restituire un puntatore a NULL e dimensione dell'array pari a zero.
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 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

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

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Prodotto`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine decrescente in base al costo (i prodotti più costosi vengono *prima* nell'array); a parità di costo, in ordine crescente in base al nome. A tal scopo, il candidato utilizzi un algoritmo di ordinamento a scelta tra quelli visti a lezione.

Per motivi ignoti accade che nel file dei clienti siano presenti clienti dal nome e/o cognome diverso, ma con lo stesso identificatore. Tali clienti sono da considerarsi un "errore", e vanno ignorati. Si definisca una funzione:

```
list eliminaDup(list clienti);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Cliente`**, restituisca in uscita una nuova lista in cui eventuali clienti con lo stesso identificatore siano stati eliminati. Ad esempio, se la lista ingresso dovesse contenere due clienti con id pari a 37, entrambi quei clienti non dovrebbero comparire nella lista restituita in uscita.

Esercizio 3 – Valutazione delle proposte (modulo banca.h/banca.c)

La banca vuole vendere più prodotti (e possibilmente i più costosi) ai clienti: ovviamente deve proporre prodotti che i clienti siano in grado di acquistare. A tal scopo deve proporre al cliente tutti i prodotti che si può permettere di acquistare: attenzione, man mano che il cliente acquista dei prodotti, il suo saldo diminuisce (perché ha già speso per alcuni prodotti).

A tal scopo si sviluppi una procedura:

```
void proposte(list clienti, Prodotto * v, int dim);
```

che, ricevuti in ingresso una lista di strutture dati di tipo **`Cliente`**, un vettore di strutture dati di tipo **`Prodotto`** e la dimensione di tale vettore, stampi a video, per ogni cliente, un insieme di proposte di prodotti. In particolare, per ogni cliente, la procedura deve:

- Scorrere l'array dei prodotti e trovare il primo prodotto più costoso che il cliente si può permettere (cioè il prodotto più costoso, il cui costo però è inferiore al saldo attuale di quel cliente),
- Stampare a video la proposta di tale prodotto,
- Chiedere al cliente se accetta di acquistarlo o meno
- Il cliente può rispondere con un 'y' o un 'n' seguito da 'invio' (tramite standard input)
    - In caso di risposta 'y', il prodotto si considera acquistato, e di conseguenza il saldo del cliente deve essere decrementato in base al costo del prodotto
    - In caso di risposta 'n', il prodotto non è acquistato e quindi il saldo del cliente resta invariato
- Ricominciare a scorrere l'array, cercando il successivo prodotto più costoso che il cliente si può permettere: si ripete l'algoritmo finché ci sono prodotti da suggerire a quel cliente specifico.

Ad esempio un output plausibile è il seguente:

```
Cliente:  Andrea Galassi Saldo Attuale: 1234.56
Vuoi acquistare il prodotto: CreditoFelice? y
Grazie! Hai acquistato "CreditoFelice", il tuo saldo attuale e' 234.56
Vuoi acquistare il prodotto: AutoNolo? y
Grazie! Hai acquistato "AutoNolo", il tuo saldo attuale e' 134.57
Vuoi acquistare il prodotto: BiciNolo? y
Grazie! Hai acquistato "BiciNolo", il tuo saldo attuale e'  44.58
```

Esercizio 4 – Proposte al cliente e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file tutte le informazioni e proponga ai clienti i prodotti. 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 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

"element.h":

```
#ifndef _CLIENTE_H
#define _CLIENTE_H

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

#define DIM_NOME 256
#define DIM_COGNOME 256
#define DIM_TEL 16
#define DIM_PRODOTTO 16

typedef struct {
	int id;
	char nome[DIM_NOME];
	char cognome[DIM_COGNOME];
	char tel[DIM_TEL];
	float saldo;
} Cliente;

typedef struct {
	int id;
	char nome[DIM_PRODOTTO];
	float costo;
} Prodotto;

typedef Cliente element;

int compare(Prodotto p1, Prodotto p2);

#endif
```

"element.c":

```
#include "element.h"


int compare(Prodotto p1, Prodotto p2) {
	int result;

	if (p1.costo > p2.costo)
		result = -1;
	else
		if (p1.costo < p2.costo)
			result = 1;
		else
			result = strcmp(p1.nome, p2.nome);
	return result;
}
```

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

Prova d'Esame 3A di Venerdì 9 Febbraio 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 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

```
void showlist(list l) {
	element temp;
	if (!empty(l)) {
		temp = head(l);
		printf("%d %s %s %s %6.2f\n",
			temp.id, temp.nome, temp.cognome, temp.tel, temp.saldo);
		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 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

"banca.h":

```
#ifndef _BANCA_H
#define _BANCA_H

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

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

// Es. 1
list leggiClienti(char * fileName);
Prodotto * leggiProd(char * fileName, int * dim);
void stampaProd(Prodotto * prod, int dim);

// Es. 2
void ordina(Prodotto * v, int dim);
list eliminaDup(list clienti);

// Es. 3
void proposte(list clienti, Prodotto * v, int dim);

#endif
```

"banca.c":

```
#include "banca.h"

list leggiClienti(char * fileName) {
	list result;
	FILE * fp;
	Cliente temp;
	char ch;
	int i;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		while (fscanf(fp, "%d", &(temp.id)) == 1) {
			i=0;
			while ((ch=fgetc(fp))!=';' && ch!='\n' && ch!=EOF && i<DIM_NOME-1) {
				temp.nome[i++] = ch;
			}
			temp.nome[i] = '\0';
			i=0;
			while ((ch=fgetc(fp))!=';' && ch!='\n' && ch!=EOF && i<DIM_COGNOME-1)
{
				temp.cognome[i++] = ch;
			}
			temp.cognome[i] = '\0';
			fscanf(fp, "%s%f", temp.tel, &(temp.saldo));
			result = cons(temp, result);
		}
		fclose(fp);
	}
```

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

Prodotto * leggiProd(char * fileName, int * dim) {
	Prodotto * result;
	Prodotto temp;
	FILE * fp;
	int i;

	result = NULL;
	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		while (fscanf(fp, "%d%s%f", &(temp.id), temp.nome, &(temp.costo)) == 3)
			*dim = *dim + 1;
		rewind(fp);
		i=0;
		result = (Prodotto*) malloc(sizeof(Prodotto) * *dim);
		while (fscanf(fp, "%d%s%f", &(temp.id), temp.nome, &(temp.costo)) == 3) {
			result[i] = temp;
			i++;
		}
		fclose(fp);
	}
	else {
		printf("Errore nell'apertura del file %s\n", fileName);
	}
	return result;
}
void stampaProd(Prodotto * prod, int dim) {
	int i;
	for (i=0; i<dim; i++) {
		printf("%d %s %6.2f\n", prod[i].id, prod[i].nome, prod[i].costo);
	}
}

// Es. 2
void  merge(Prodotto v[], int i1, int i2, int fine, Prodotto 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(Prodotto v[], int first, int last, Prodotto 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(Prodotto * v, int dim) {
	Prodotto * temp = (Prodotto*) malloc(sizeof(Prodotto) * dim);
	mergeSort(v, 0, dim-1, temp);
	free(temp);
}
// si noti che la conta(…) considera ripetuti gli elementi in base al solo id; una
soluzione altrettanto valida sarebbe stata verificare anche nome e cognome…
int conta(Cliente c, list elenco) {
	int result = 0;
	while (!empty(elenco)) {
		if (c.id == head(elenco).id)
			result++;
		elenco = tail(elenco);
	}
	return result;
}
list eliminaDup(list clienti) {
	list temp = clienti;
	list result = emptylist();
	while (!empty(temp)) {
		if (conta(head(temp), clienti)==1)
			result = cons(head(temp), result);
		temp = tail(temp);
	}
	return result;
}

void proposte(list clienti, Prodotto * v, int dim) {
	Cliente temp;
	int i;
	char answer;
	ordina(v, dim);
	while (!empty(clienti)) {
		temp = head(clienti);
		printf("Cliente: %s %s\n", temp.nome, temp.cognome);
		printf("Saldo Attuale: %6.2f\n", temp.saldo);
		for (i=0; i<dim; i++) {
			if (v[i].costo<temp.saldo) {
				printf("Vuoi acquistare il prodotto: %s? ", v[i].nome);
				scanf("%c%*c", &answer);
				if (answer == 'y') {
					temp.saldo = temp.saldo - v[i].costo;
					printf("Grazie! Hai acquistato \"%s\", il tuo saldo
attuale e' %6.2f\n", v[i].nome, temp.saldo);
				}
			}
		}
		printf("-------------------\n");
		clienti=tail(clienti);
	}
	return;
}
```

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

Prova d'Esame 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{ // Es. 1
		list elenco;
		Prodotto * prod;
		int dim;
		elenco = leggiClienti("clienti.txt");
		showlist(elenco);
		printf("\n\n");
		prod = leggiProd("prodotti.txt", &dim);
		stampaProd(prod, dim);
		printf("\n\n");
		free(prod);
		freelist(elenco);
	}
	{	// Es. 2
		list elenco;
		list filtrato;
		Prodotto * prod;
		int dim;
		prod = leggiProd("prodotti.txt", &dim);
		ordina(prod, dim);
		stampaProd(prod, dim);
		elenco = leggiClienti("clienti.txt");
		filtrato = eliminaDup(elenco);
		showlist(filtrato);
		printf("\n\n");
		free(prod);
		freelist(elenco);
		freelist(filtrato);
	}
	{	// Es. 3 & 4
		list elenco;
		list filtrato;
		Prodotto * prod;
		int dim;
		prod = leggiProd("prodotti.txt", &dim);
		elenco = leggiClienti("clienti.txt");
		filtrato = eliminaDup(elenco);
		printf("\n\n");
		proposte(filtrato, prod, dim);
		free(prod);
		freelist(elenco);
		freelist(filtrato);
	}

	return 0;
}
```

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

Prova d'Esame 3A di Venerdì 9 Febbraio 2018 – tempo a disposizione 2h

"clienti.txt":

```
37 Marco;Lipparini;+390519898989 1000.00
11 Federico Domenico;Chesani;+39051121212 5600.79
13 Paola Chiara;Mello;+390512323232 8765.43
17 Daniela Giovanna;Loreti;+393434343 567.89
23 Luca;Cattelani;+390514545454 899.54
29 Andrea;Galassi;+390515656565 1234.56
37 Marco;Lippi;+390517676767 1000.00
```

"prodotti.txt":

```
60 CreditoFelice 1000.00 80 AssicCasa 1500.00
90 BitCoinSicuri 5999.99


30 InvestiUSA 599.00 35 InvestiCanada 499.00 37 InvestiPanama 699.00
39 InvestiBrasil 799.00

40 AutoNolo 99.99 45 BiciNolo 89.99
```