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

Prova d'Esame 1A di Giovedì 14 Gennaio 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 hotel registra le presenze dei clienti tramite un file di testo. In particolare, per ogni soggiorno effettuato da un cliente, vengono registrate su una riga le seguenti informazioni: **l'identificatore** del cliente (un intero); a seguire, separata da uno spazio, la **data** di inizio del soggiorno (nel formato aaaa/mm/gg, tre interi); a seguire, separata da uno spazio, la **durata** in giorni del soggiorno (un intero); infine, ancora separata da uno spazio, la descrizione del **tipo** di camera usata (una stringa di al più 1023 caratteri utili, senza spazi). Tutte le righe sono terminate da un carattere 'newline', tranne l'ultima, che a volte lo è e a volte no.
In un secondo file di testo vengono memorizzate invece le tariffe, che variano a seconda del tipo di camera e del periodo dell'anno. Su ogni riga sono memorizzate nel seguente ordine: la descrizione del **tipo** di camera (memorizzata come sopra); separata da uno spazio, la data di **inizio** del periodo di validità prezzi (sempre nel formato aaaa/mm/gg); separata da uno spazio, la data di **fine** del periodo di validità (formato aaaa/mm/gg); e infine, separato da uno spazio, il **prezzo** giornaliero relativo al tipo di camera (un float).
Si vedano, a titolo di esempio, i file di testo "clienti.txt" e "tariffe.txt" forniti nello StartKit.

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

Si definisca un'opportuna struttura dati **`Data`**, al fine di rappresentare una data definita come giorno, mese e anno (tre interi). Si definisca un'opportuna struttura dati **`Soggiorno`**, al fine di rappresentare le informazioni relative ad un soggiorno (id del cliente, data di inizio, durata, e tipo di camera). Si definisca infine la struttura dati **`Tariffa`**, al fine di rappresentare i dati relativi ad una singola tariffa (tipo della camera, data di inizio e data di fine, prezzo).
Si definisca la funzione:

```
Soggiorno * leggiSoggiorni(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente le informazioni relative ai soggiorni, legga da tale file tali informazioni, e le restituisca tramite un array allocato dinamicamente di strutture dati di tipo **`Soggiorno`**. Tramite il parametro **`dim`** passato per riferimento la funzione deve restituire la dimensione dell'array allocato dinamicamente. L'array deve avere la dimensione minima possibile. Si assuma per semplicità che il file di input sia sempre ben formato secondo le specifiche date sopra. 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 leggiTariffe(char* fileName);
```

che, ricevuto in ingresso il nome di un file di testo contenente le tariffe, restituisca una lista di strutture dati di tipo **`Tariffa`**, contenente tutte 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 stampaSoggiorni(Soggiorno * v, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Soggiorno`**, e la sua dimensione **`dim`**, stampi a video il contenuto di tali strutture dati.
Si definisca la procedura ricorsiva:

```
void stampaTariffe(list elenco);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Tariffa`**, ne stampi a video il contenuto.
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 1A di Giovedì 14 Gennaio 2016 – tempo a disposizione 2h

Esercizio 2 – Ordinamento (moduli element.h/.c e clienti.h/.c)

Il candidato definisca una funzione:

```
int compareData(Data d1, Data d2);
```

che restituisca un valore minore di zero, pari a zero, o maggiore di zero, a seconda che la data **d1** preceda, sia uguale o segua la data **d2**.

Il candidato definisca una procedura:

```
void ordinaSoggiorni(Soggiorno * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **Soggiorno** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in maniera crescente in base all'id del cliente; a parità di id, in ordine crescente in base alla data di inizio del soggiorno. A tal fine, il candidato utilizzi l'algoritmo "naive 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.

Esercizio 3 –Calcolo dei totali (modulo clienti.h/clienti.c)

Si vuole realizzare una funzionalità che permetta, dato uno specifico cliente, di calcolare il totale dovuto. A tal scopo, il candidato definisca la funzione:

```
float costoSoggiorno(Soggiorno uno, list elencoTariffe);
```

che, ricevuti come parametri un soggiorno e l'elenco delle tariffe, restituisca il costo legato a tale soggiorno. Per determinare la tariffa giusta, si consideri ovviamente il tipo di camera e la sola data di inizio del soggiorno. Qualora un soggiorno abbia una durata riferibile a due tariffe diverse, si consideri solo e comunque la data di inizio soggiorno. Qualora non sia possibile trovare una tariffa relativa alla data di inizio del soggiorno, la funzione deve restituire come costo il valore zero.

Si definisca poi la funzione:

```
float totale(int idCliente, list elencoTariffe, Soggiorno * elencoSog, int dim);
```

che, ricevuto come parametro l'identificatore unico di un cliente specifico, la lista di strutture dati di tipo **Tariffa**, e il vettore dei soggiorni di tutti i clienti (con la sua dimensione), provveda a calcolare il totale attribuibile al solo cliente specificato. Il totale è dato dalla somma del costo di ogni singolo soggiorno riferibile al cliente.

Esercizio 4 – Stampa del totale riferibile ad un cliente, 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 soggiorni memorizzati nel file "clienti.txt". Il programma chieda poi all'utente di specificare un cliente (tramite il suo id), e calcoli e stampi a video il totale dovuto da tale cliente in riferimento ai soggiorni effettuati.

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 1A di Giovedì 14 Gennaio 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

typedef struct {
	int giorno;
	int mese;
	int anno;
} Data;

typedef struct {
	int id;
	Data inizio;
	int durata;
	char desc[1024];
} Soggiorno;

typedef struct {
	char desc[1024];
	Data inizio;
	Data fine;
	float prezzo;
} Tariffa;

typedef Tariffa element;

int compareData(Data d1, Data d2);
int compareSoggiorno(Soggiorno s1, Soggiorno s2);

#endif
```

"element.c":

```
#include "element.h"

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

int compareSoggiorno(Soggiorno s1, Soggiorno s2) {
	if (s1.id != s2.id)
		return s1.id - s2.id;
	else
		return compareData(s1.inizio, s2.inizio);
}
```

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

Prova d'Esame 1A di Giovedì 14 Gennaio 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);
//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 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 1A di Giovedì 14 Gennaio 2016 – tempo a disposizione 2h

"clienti.h":

```
#ifndef _CLIENTI_H
#define _CLIENTI_H

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

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

Soggiorno * leggiSoggiorni(char * fileName, int * dim);
list leggiTariffe(char* fileName);
void stampaSoggiorni(Soggiorno * v, int dim);
void stampaTariffe(list elenco);

void ordinaSoggiorni(Soggiorno * v, int dim);

float costoSoggiorno(Soggiorno uno, list elencoTariffe);
float totale(int idCliente, list elencoTariffe, Soggiorno * elencoSog, int dim);

#endif
```

"clienti.c":

```
#include "clienti.h"

Soggiorno * leggiSoggiorni(char * fileName, int * dim) {
	FILE * fp;
	Soggiorno * result = NULL;
	Soggiorno temp;
	int count;
	int i;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Problemi nell'apertura del file: %s\n", fileName);
		return result;
	}

	// conto i soggiorni nel file...
	count = 0;
	while(fscanf(fp, "%d %d/%d/%d %d %s",
		&temp.id, &(temp.inizio.anno), &(temp.inizio.mese), &(temp.inizio.giorno),
		&(temp.durata), temp.desc) == 6)
		count++;

	// alloco memoria e "riavvolgo" il file
	result = (Soggiorno *) malloc(sizeof(Soggiorno) * count);
	rewind(fp);

	// lettura vera e propria
	i = 0;
```

```
	while(fscanf(fp, "%d %d/%d/%d %d %s", &(temp.id), &(temp.inizio.anno),
&(temp.inizio.mese), &(temp.inizio.giorno), &(temp.durata), temp.desc) == 6) {
		result[i] = temp;
		i++;
	}

	fclose(fp);
	*dim = i;
	return result;
}

list leggiTariffe(char* fileName) {
	list result;
	FILE * fp;
	Tariffa temp;

	result = emptylist();

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Problemi nell'apertura del file: %s\n", fileName);
		return result;
	}
	while (fscanf(fp, "%s %d/%d/%d %d/%d/%d %f",
		temp.desc, &(temp.inizio.anno), &(temp.inizio.mese), &(temp.inizio.giorno),
		&(temp.fine.anno), &(temp.fine.mese), &(temp.fine.giorno),
		&(temp.prezzo)) == 8)
			result = cons(temp, result);

	fclose(fp);
	return result;
}

void stampaSoggiorni(Soggiorno * v, int dim) {
	int i;
	for (i=0; i<dim; i++)
		printf("%d %d/%d/%d %d %s\n", v[i].id, v[i].inizio.anno, v[i].inizio.mese,
v[i].inizio.giorno, v[i].durata, v[i].desc);
		return;
}

void stampaTariffe(list elenco) {
	Tariffa temp;
	if (empty(elenco)) {
		//do nothing
	}
	else {
		temp = head(elenco);
		printf("%s %d/%d/%d %d/%d/%d %f\n",
			temp.desc, temp.inizio.anno, temp.inizio.mese, temp.inizio.giorno,
			temp.fine.anno, temp.fine.mese, temp.fine.giorno,
			temp.prezzo );
		stampaTariffe(tail(elenco));
	}
	return;
}
```

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

Prova d'Esame 1A di Giovedì 14 Gennaio 2016 – tempo a disposizione 2h

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

int compareSoggiorno(Soggiorno s1, Soggiorno s2) {
	int result;
	result = s1.id - s2.id;
	if (result == 0)
		result = compareData(s1.inizio, s2.inizio);
	return result;
}

int trovaPosMax(Soggiorno v[], int n){
	int i, posMax=0;
	for (i=1; i<n; i++)
		if (compareSoggiorno(v[posMax], v[i]) < 0)
			posMax=i;
	return posMax;
}

void naiveSort(Soggiorno v[], int n){
	int p;
	while (n>1) {
		p = trovaPosMax(v,n);
		if (p<n-1) scambia(&v[p],&v[n-1]);
		n--;
	}
}

void ordinaSoggiorni(Soggiorno * v, int dim) {
	naiveSort(v, dim);
}

float costoSoggiorno(Soggiorno uno, list elencoTariffe) {
	int trovato = 0;
	Tariffa temp;

	while (!empty(elencoTariffe) && ! trovato) {
		temp = head(elencoTariffe);
		if (strcmp(uno.desc, temp.desc) == 0
			&& compareData(uno.inizio, temp.inizio)>=0
			&& compareData(uno.inizio, temp.fine)<=0) {
				trovato = 1;
			}
		else
			elencoTariffe = tail(elencoTariffe);
	}
	if (trovato == 1)
		return temp.prezzo * uno.durata;
	else
		return 0.0F;
}

float totale(int idCliente, list elencoTariffe, Soggiorno * elencoSog, int dim) {
	int i;
```

```
        float prezzoTotale;

        prezzoTotale = 0.0F;
        for (i=0; i<dim; i++) {
               if (idCliente == elencoSog[i].id)
                      prezzoTotale = prezzoTotale + costoSoggiorno(elencoSog[i],
elencoTariffe);
        }
        return prezzoTotale;
}
```

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

Prova d'Esame 1A di Giovedì 14 Gennaio 2016 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>

#include "element.h"
#include "clienti.h"

int main()
{
	// Es. 1
	{
		int dim;
		Soggiorno * elencoSoggiorni;
		list elencoTariffe;

		elencoSoggiorni = leggiSoggiorni("clienti.txt", &dim);
		stampaSoggiorni(elencoSoggiorni, dim);

		elencoTariffe = leggiTariffe("tariffe.txt");
		stampaTariffe(elencoTariffe);

		free(elencoSoggiorni);
		freelist(elencoTariffe);
	}

	// Es. 2
	{
		int dim;
		Soggiorno * elencoSoggiorni;

		printf("\n\n\n");
		elencoSoggiorni = leggiSoggiorni("clienti.txt", &dim);
		stampaSoggiorni(elencoSoggiorni, dim);

		printf("\n\n\n");
		ordinaSoggiorni(elencoSoggiorni, dim);
		stampaSoggiorni(elencoSoggiorni, dim);

		free(elencoSoggiorni);
	}

	// Es. 3 && 4
	{
		int dim;
		Soggiorno * elencoSoggiorni;
		list elencoTariffe;
		int id;

		elencoSoggiorni = leggiSoggiorni("clienti.txt", &dim);
		elencoTariffe = leggiTariffe("tariffe.txt");

		printf("\n\n\n");
		stampaSoggiorni(elencoSoggiorni, dim);
		printf("\nInserire id cliente: ");
		scanf("%d", &id);
		printf("Totale dovuto dal cliente %d: %f\n", id,
				totale(id, elencoTariffe, elencoSoggiorni, dim));
```

```
            free(elencoSoggiorni);
            freelist(elencoTariffe);
        }


        return 0;
}
```

"clienti.txt":

```
12 2015/01/03 4 cameraDoppia
17 2015/02/04 5 cameraSingola
17 2015/03/05 3 cameraSingola
12 2015/04/06 3 cameraDoppia
12 2015/05/07 6 cameraDoppia
12 2015/06/06 1 cameraDoppia
19 2015/07/10 1 cameraDoppiaUsoSingola
12 2015/08/03 21 cameraDoppia
```

"tariffe.txt":

```
cameraSingola 2015/01/01 2015/06/30 100.0
cameraSingola 2015/07/01 2015/12/31 110.0
cameraDoppia 2015/01/01 2015/06/30 200.0
cameraDoppia 2015/07/01 2015/12/31 220.0
cameraDoopiaUsoSingola 2015/01/01 2015/06/30 150.0
cameraDoppiaUsoSingola 2015/07/01 2015/12/31 160.0
```