Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – 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 compagnia aerea ha deciso di sviluppare un sistema informatico per la verifica delle procedure di check-in online, e l'afferenza dei passeggeri nei voli aerei.
In un primo file di testo la compagnia memorizza, in ordine casuale, la lista dei biglietti acquistati per tutti i voli: su ogni riga registra prima l'identificativo del **volo** (una stringa di 6 caratteri utili, senza spazi); a seguire, separato da uno spazio, l'identificativo del **biglietto** aereo relativo a quel volo (una stringa alfanumerica di al più 31 caratteri utili, senza spazi).
In un secondo file di testo la compagnia aerea memorizza le seguenti informazioni: l'identificativo del **biglietto**; separato da un carattere ';' il **nome e cognome** del viaggiatore (una stringa di al più 1023 caratteri, contenente spazi); infine, ancora separato da un carattere ';', un carattere riguardante l'operazione di **check-in** online: 'y' indica che è già stato fatto il check-in, 'n' indica che il check-in è ancora da fare.
Si vedano, a titolo di esempio, i file "prenotazioni.txt" e "viaggiatori.txt" forniti nello StartKit.

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

Si definiscano opportune strutture dati **`Prenotazione`** (per memorizzare id di un volo e id di un biglietto), e **`Viaggiatore`** (per memorizzare id di un biglietto, nome del viaggiatore, e check-in effettuato si/no).

Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file, legga da tale file tutte le informazioni relative alle prenotazioni e le restituisca tramite un array di strutture dati di tipo **`Prenotazione`** 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 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 definisca la procedura:

```
void stampaPrenotazioni(Prenotazione * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Prenotazione`**, e la sua dimensione **`dim`**, stampi a video le informazioni riguardo le prenotazioni registrate nel file.

Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file, legga da tale file tutte le informazioni relative ai viaggiatori e le restituisca tramite una lista di strutture dati di tipo **`Viaggiatore`**. 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.

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, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – tempo a disposizione 2h

Esercizio 2 –Ordinamenti (moduli element.h/.c e voli.h/.c)

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Prenotazione`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine lessicografico crescente in base all'identificatore del volo; in caso di uguaglianza, in ordine lessicografico crescente in base al nome del viaggiatore. A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "merge sort" visto a lezione.

Si definisca poi una funzione:

```
list ordinaLista(list viaggiatori);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Viaggiatore`**, restituisca una nuova lista di strutture dati di tipo **`Viaggiatore`**, ma ordinata secondo il seguente criterio: prima devono venire tutti coloro che hanno già effettuato il check-in; a seguire, tutti coloro che non l'hanno ancora fatto. A parità di check-in, la lista dovrà essere ordinata in ordine lessicografico in base al nome del viaggiatore.

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.

Esercizio 3 – Verifica del check-in e viaggiatori mancanti (modulo voli.h/voli.c)

La compagnia aerea vuole conoscere tutti i viaggiatori che dovranno prendere parte ad un dato volo. A tal scopo, si definisca una funzione:

```
Viaggiatore estraiSingolo(list viaggiatori, char * biglietto);
```

che ricevuti come parametri una lista di strutture dati di tipo **`Viaggiatore`**, e l'identificatore unico di un biglietto, restituisca in uscita la struttura dati di tipo **`Viaggiatore`** relativa al biglietto specificato. Qualora il biglietto non sia presente in lista, la funzione deve restituire una struttura dati col campo nome pari alla stringa "NULL".

Si definisca una funzione:

```
list estraiViaggiatori(Prenotazione * v, int dim, list viaggiatori, char *
                                                        idVolo);
```

che ricevuti come parametri un vettore di strutture dati di tipo **`Prenotazione`** e la sua dimensione, una lista di strutture dati di tipo **`Viaggiatore`**, e l'identificatore unico di un volo, restituisca in uscita la lista di strutture **`Viaggiatore`** relative al volo specificato. La funzione dovrà trovare tutti i biglietti elencati nell'array e relativi al volo specificato, e dovrà estrarre la lista delle strutture dati di tipo **`Viaggiatore`** relative ai biglietti.

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

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file i dati relativi ai voli e ai viaggiatori. Il programma poi chieda all'utente di specificare il codice di un volo, e stampi a video tutti i viaggiatori relativi a quel volo.

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, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – tempo a disposizione 2h

"element.h":

```
#include <string.h>

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_VOLO 7
#define DIM_BIGLIETTO 32
#define DIM_NOME 1024

typedef struct {
	char volo[DIM_VOLO];
	char biglietto[DIM_BIGLIETTO];
} Prenotazione;


typedef struct {
	char biglietto[DIM_BIGLIETTO];
	char nome[DIM_NOME];
	char checkin;
} Viaggiatore;


typedef Viaggiatore element;

int comparePrenotazione(Prenotazione p1, Prenotazione p2);
int compareViaggiatore(Viaggiatore v1, Viaggiatore v2);

#endif
```

"element.c":

```
#include "element.h"


int comparePrenotazione(Prenotazione p1, Prenotazione p2) {
	int result;
	result = strcmp(p1.volo, p2.volo);
	if (result==0)
		result = strcmp(p1.biglietto, p2.biglietto);
	return result;
}


int compareViaggiatore(Viaggiatore v1, Viaggiatore v2) {
	int result;

	result = v2.checkin - v1.checkin;
	if (result == 0) {
		result = strcmp(v1.nome, v2.nome);
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – 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 showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%s %s %c\n",
                     temp.biglietto, temp.nome, temp.checkin);
              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;
}
*/


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


list insord_p(element el, list l) {
```

```
        list pprec, patt = l, paux;
        int trovato = 0;

        while (patt!=NULL && !trovato) {
                if (compareViaggiatore(el, patt->value)<0)
                        trovato = 1;
                else {
                        pprec = patt;
                        patt = patt->next;
                }
        }
        paux = (list) malloc(sizeof(item));
        paux->value = el;
        paux->next = patt;
        if (patt==l)
                return paux;
        else {
                pprec->next = paux;
                return l;
        }
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – tempo a disposizione 2h

"voli.h":

```
#ifndef _VOLI_H
#define _VOLI_H

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

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


// Es. 1
Prenotazione * leggiTutti(char * fileName, int * dim);
void stampaPrenotazioni(Prenotazione * v, int dim);
list leggiViaggiatori(char * fileName);

// Es. 2
void ordina(Prenotazione * v, int dim);
list ordinaLista(list viaggiatori);

// Es. 3
Viaggiatore estraiSingolo(list viaggiatori, char * biglietto);
list estraiViaggiatori(Prenotazione * v, int dim, list viaggiatori, char * idVolo);

#endif
```

"voli.c":

```
#include "voli.h"

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


	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		count = 0;
		while (fscanf(fp, "%s%s", temp.volo, temp.biglietto) == 2)
			count++;
		rewind(fp);

		i=0;
		result = (Prenotazione*) malloc(sizeof(Prenotazione) * count);
		while (fscanf(fp, "%s%s", temp.volo, temp.biglietto) == 2 && i<count) {
			result[i] = temp;
			i++;
		}
		*dim = i;
		fclose(fp);
	}
```

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

void stampaPrenotazioni(Prenotazione * v, int dim) {
	int i;
	for (i=0; i<dim; i++) {
		printf("%s %s\n", v[i].volo, v[i].biglietto);
	}
	return;
}

int readField(char buffer[], int dimBuffer, char sep, FILE *f) {
	int i = 0;
	char ch = fgetc(f);
	while (ch=='\n' || ch=='\r') {
		ch = fgetc(f);
	}
	while (i<dimBuffer-1 && ch != sep && ch != 10 && ch != EOF) {
		buffer[i] = ch;
		i++;
		ch = fgetc(f);
	}
	buffer[i] = '\0';
	return ch;
}

list leggiViaggiatori(char * fileName) {
	list result;
	FILE * fp;
	Viaggiatore temp;
	char ch;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		do {
			ch = readField(temp.biglietto, DIM_BIGLIETTO, ';', fp);
			if (ch==';') {
				readField(temp.nome, DIM_NOME, ';', fp);
				temp.checkin = fgetc(fp);
				result = cons(temp, result);
			}
		} while (ch == ';');
		fclose(fp);
	}
	else {
		printf("Errore nell'apertura del file %s\n", fileName);
	}



	return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – tempo a disposizione 2h

```
void  merge(Prenotazione v[], int i1, int i2, int fine, Prenotazione vout[]) {
       int i=i1, j=i2, k=i1;
       while ( i <= i2-1 &&  j <= fine ) {
              if (comparePrenotazione(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(Prenotazione v[], int first, int last, Prenotazione 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(Prenotazione * v, int dim) {
       Prenotazione * temp;

       temp = (Prenotazione*) malloc(sizeof(Prenotazione)*dim);
       mergeSort(v, 0, dim-1, temp);
       free(temp);
}

list ordinaLista(list viaggiatori) {
       list result = emptylist();
       Viaggiatore temp;

       while (!empty(viaggiatori)) {
              temp = head(viaggiatori);
              result = insord_p(temp, result);
              viaggiatori = tail(viaggiatori);
       }
       return result;
}




Viaggiatore estraiSingolo(list viaggiatori, char * biglietto) {
       Viaggiatore result;
       int trovato = 0;

       strcpy(result.nome, "NULL");
```

```
        strcpy(result.biglietto, biglietto);
        result.checkin = 'n';
        while (!empty(viaggiatori) && !trovato) {
                if (strcmp(biglietto, head(viaggiatori).biglietto)==0) {
                        result = head(viaggiatori);
                        trovato = 1;
                }
                viaggiatori = tail(viaggiatori);
        }
        return result;
}

list estraiViaggiatori(Prenotazione * v, int dim, list viaggiatori, char * idVolo) {
        list result;
        int i;

        result = emptylist();
        for (i=0; i<dim; i++) {
                if (strcmp(idVolo, v[i].volo)==0) {
                        result = cons(estraiSingolo(viaggiatori, v[i].biglietto), result);
                }
        }
        return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 4A di Giovedì 15 Giugno 2017 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "list.h"
#include "voli.h"

int main() {
	// Es. 1
	{
		Prenotazione * p;
		int dim;
		list v;

		p = leggiTutti("prenotazioni.txt", &dim);
		if (dim>0)
			stampaPrenotazioni(p, dim);
		v = leggiViaggiatori("viaggiatori.txt");
		showlist(v);

		free(p);
		freelist(v);
	}

	// Es. 2
	{
		Prenotazione * p;
		int dim;
		list v;
		list v_ord;

		p = leggiTutti("prenotazioni.txt", &dim);
		if (dim>0) {
			ordina(p, dim);
			stampaPrenotazioni(p, dim);
		}
		v = leggiViaggiatori("viaggiatori.txt");
		v_ord = ordinaLista(v);
		showlist(v_ord);

		free(p);
		freelist(v);
		freelist(v_ord);
	}


	// Es. 3 && 4
	{
		Prenotazione * p;
		int dim;
		list v;
		list v2;
		char volo[DIM_VOLO];

		p = leggiTutti("prenotazioni.txt", &dim);
		v = leggiViaggiatori("viaggiatori.txt");

		printf("Inserire volo: ");
```

```
            scanf("%s", volo);
            v2 = estraiViaggiatori(p, dim, v, volo);
            showlist(v2);

            free(p);
            freelist(v);
            freelist(v2);
        }
        return 0;
}
```

"prenotazioni.txt":

```
AZ2456 BQQ4YT
AZ2456 GHT67U
AZ1122 FTRE43
AZ2456 KL3EEF
AZ2456 GH4ERT
AZ2456 XX55YY
```

"viaggiatori.txt"

```
BQQ4YT;Federico Chesani;y
GHT67U;Chiara Chesani;y
FTRE43;Paola Mello;y
KL3EEF;Francesco Chesani;y
GH4ERT;Daniela Loreti;n
```