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

Prova d'Esame 6A di Giovedì 8 Settembre 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 famoso giornalista salva sul proprio computer, in un file di testo, le dichiarazioni fatte da persone famose. In particolare, memorizza su ogni riga del file una dichiarazione, nel seguente modo: all'inizio del file salva la **data** in cui la dichiarazione è stata fatta (tre interi, formato "gg/mm/aaaa"); a seguire, separato da uno spazio, il **nome** della persona famosa (una stringa di al più 255 caratteri, senza spazi); infine, ancora separata da uno spazio, il **testo** della dichiarazione stessa (una stringa di al più 4095 caratteri, senza spazi). In particolare il giornalista, che ha avuto un passato come studente di Ingegneria Informatica, memorizza il testo della dichiarazione con un piccolo accorgimento: tutte le volte che ha incontrato il carattere "spazio", lo ha sostituito col carattere "underscore". Quindi ad esempio un testo: "Domani risolveremo i problemi" viene memorizzato nel file come "Domani_risolveremo_i_problemi".
Non è noto a priori quante righe (cioè, quante dichiarazioni) siano memorizzate nel file.
Si veda, a titolo di esempio, il file di testo "dichiarazioni.txt" fornito nello StartKit.

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

Si definiscano opportune strutture dati **`Data`**, e **`Dichiarazione`**, al fine di rappresentare una dichiarazione fatta da una persona famosa (data, nome della persona, e testo della dichiarazione).
Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file, legga da tale file le informazioni relative alle dichiarazioni e le restituisca memorizzate in una lista di strutture dati di tipo **`Dichiarazione`**. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, e restituire una lista vuota.

Il giornalista memorizza il testo delle dichiarazioni sostituendo ad ogni spazio il carattere underscore. Però in fase di stampa dei dati memorizzati sul file, desidera avere il testo nella forma originale (cioè con gli spazi e non con gli underscore).
A tal scopo, si definisca la funzione:

```
Dichiarazione mettiSpazi(Dichiarazione source);
```

che, ricevuta in ingresso una struttura dati di tipo **`Dichiarazione`**, restituisca una nuova struttura dati di tipo **`Dichiarazione`** dove nel testo della dichiarazione ogni carattere underscore '_' è stato sostituito da un carattere di spazio.
Si definisca poi la funzione:

```
list trasforma(list aList);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Dichiarazione`**, restituisca in uscita una nuova lista, dove i testi delle dichiarazioni siano stati opportunamente trasformati: al posto dei caratteri underscore, dovranno comparire degli spazi.
Si definisca la procedura:

```
void stampaDich(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Dichiarazione`**, stampi a video le dichiarazioni. La procedura deve fare in modo che, a prescindere dalle dichiarazioni fornite in ingresso, stampi comunque i testi di queste dichiarazioni con gli spazi e non con gli underscore.
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, 2015/2016 – Modulo 2

Prova d'Esame 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

Esercizio 2 – Filtraggio e ordinamento (moduli element.h/.c e dich.h/.c)

Al fine di poter comprendere le dichiarazioni, il giornalista necessita di poter filtrare le dichiarazioni in base ad una keyword da specificare, e di poter ordinare le dichiarazioni in base ad alcuni criteri. A tal scopo il candidato definisca una funzione:

```
Dichiarazione * filtra(list elenco, char * keyword, int * dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Dichiarazione`** ed una parola chiave (**`keyword`**) da cercare, restituisca in uscita un vettore contenente tutte le strutture dati di tipo **`Dichiarazione`**, nel cui campo "nome" o nel campo "testo" vi sia presente la keyword specificata. Il vettore restituito dovrà essere stato allocato dinamicamente (non necessariamente della dimensione minima), e tramite il parametro **`dim`** passato per riferimento, si dovrà restituire la dimensione di tale vettore. Allo scopo di ricercare la stringa keyword nei vari campi, si ricorda l'esistenza della funzione di libreria:
`char * strstr(char * source, char * sub)`, che restituisce un puntatore alla prima occorrenza di **`sub`** in **`source`**, o **`NULL`** altrimenti.

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Dichiarazione`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine alfabetico (lessicografico) rispetto al nome del personaggio che ha rilasciato la dichiarazione; a parità di nome, in base alla data della dichiarazione, con le dichiarazioni più recenti che compaiono prima nel vettore. A tal scopo, il candidato utilizzi un algoritmo di ordinamento a scelta tra quelli visti a lezione.

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 – Identificazione dei video più visti ch econtengono una certa keyword (modulo video.h/video.c)

Il giornalista necessita di sapere chi sia la persona famosa che ha fatto più dichiarazioni su un certo argomento. Si definisca quindi una funzione:

```
char * commenti(list elenco, char * keyword);
```

che ricevuti come parametri una lista di strutture dati di tipo **`Dichiarazione`**, e una stringa rappresentante una parola chiave da cercare, restituisca un puntatore al nome della persona che ha fatto più dichiarazioni riguardanti la keyword specificata. Qualora non vi siano proprio dichiarazioni relative alla keyword specificata, la funzione deve restituire un puntatore a NULL.

Esercizio 4 Conteggio dei video che contengono una certa keyword nella descrizione, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che chieda all'utente di specificare una keyword, e stampi a video il nome della persona famosa che ha fatto più dichiarazioni relative alla keyword specificata.

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 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

#include <string.h>

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

typedef struct {
	Data data;
	char nome[256];
	char testo[4096];
} Dichiarazione;


typedef Dichiarazione element;

int compare(Dichiarazione d1, Dichiarazione d2);

#endif
```

"element.c":

```
#include "element.h"


int compare(Dichiarazione d1, Dichiarazione d2) {
	int result;

	result = strcmp(d1.nome, d2.nome);
	if (result == 0) {
		result = d2.data.anno - d1.data.anno;
		if (result == 0) {
			result = d2.data.mese - d1.data.mese;
			if (result == 0) {
				result = d2.data.giorno - d1.data.giorno;
			}
		}
	}
	return result;
}
```

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

Prova d'Esame 6A di Giovedì 8 Settembre 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 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);
}

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 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

"dich.h":

```
#ifndef _DICH_H
#define _DICH_H

#include <stdio.h>

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

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


list leggiDich(char * fileName);
Dichiarazione mettiSpazi(Dichiarazione source);
list trasforma(list aList);
void stampaDich(list elenco);

Dichiarazione * filtra(list elenco, char * keyword, int * dim);
void ordina(Dichiarazione * v, int dim);

char * commenti(list elenco, char * keyword);

#endif
```

"dich.c":

```
#include "dich.h"


list leggiDich(char * fileName) {
	list result;
	FILE * fp;
	Dichiarazione temp;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file: %s\n", fileName);
	}
	else {
		while (fscanf(fp, "%d/%d/%d %s %s",
			&(temp.data.giorno),
			&(temp.data.mese),
			&(temp.data.anno),
			temp.nome,
			temp.testo) == 5) {
				result = cons(temp, result);
		}
		fclose(fp);
	}
	return result;
}


Dichiarazione mettiSpazi(Dichiarazione source) {
```

```
	Dichiarazione dest;
	int i;

	i=0;
	dest = source;
	while (dest.testo[i] != '\0') {
		if (dest.testo[i] == '_')
			dest.testo[i] = ' ';
		i++;
	}
	return dest;
}

list trasforma(list aList) {
	list result;

	result = emptylist();
	while (!empty(aList)) {
		result = cons(mettiSpazi(head(aList)), result);
		aList = tail(aList);
	}
	return result;
}

void stampaDich(list elenco) {
	list temp;
	list temp2;
	Dichiarazione d;

	temp = trasforma(elenco);
	temp2 = temp;
	while (!empty(temp)) {
		d = head(temp);
		printf("%d/%d/%d %s %s\n",
			d.data.giorno,
			d.data.mese,
			d.data.anno,
			d.nome,
			d.testo);
		temp = tail(temp);
	}
	freelist(temp2);
}


Dichiarazione * filtra(list elenco, char * keyword, int * dim) {
	Dichiarazione * result;
	int length;
	list temp;

	length = 0;
	temp = elenco;
	while(!empty(temp)) {
		length = length + 1;
		temp = tail(temp);
	}
```

```
	result = (Dichiarazione *) malloc(sizeof(Dichiarazione) * length);
	*dim = 0;
	while (!empty(elenco)) {
		if (strstr(head(elenco).nome, keyword) != NULL
			|| strstr(head(elenco).testo, keyword) != NULL) {
				result[*dim] = head(elenco);
				*dim = *dim + 1;
		}
		elenco = tail(elenco);
	}
	return result;
}



void  merge(Dichiarazione v[], int i1, int i2, int fine, Dichiarazione 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(Dichiarazione v[], int first, int last, Dichiarazione 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(Dichiarazione * v, int dim) {
	Dichiarazione * temp;
	temp = (Dichiarazione *) malloc(sizeof(Dichiarazione) * dim);
	mergeSort(v, 0, dim-1, temp);
	free(temp);

}



char * commenti(list elenco, char * keyword) {
	char * result = NULL;
	char * current = NULL;

	Dichiarazione * temp;
	int dim;

	int count;
```

```
        int max;
        int i;

        temp = filtra(elenco, keyword, &dim);
        if (dim>0) {
                ordina(temp, dim);
                result = (char *) malloc(sizeof(char) * 256);

                current = temp[0].nome;
                count = 1;
                max = 0;
                for (i=1; i<dim; i++) {
                        if (!strcmp(current, temp[i].nome))
                                count++;
                        else {
                                if (count > max) {
                                        strcpy(result, current);
                                        max = count;
                                }
                                current = temp[i].nome;
                                count = 1;
                        }
                }
                free(temp);
        }
        return result;
}
```

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

Prova d'Esame 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{ // Es. 1
		list elenco;
		elenco = leggiDich("dichiarazioni.txt");
		stampaDich(elenco);
		freelist(elenco);
	}
	{ // Es. 2
		list elenco;
		Dichiarazione * vett;
		int dim;
		int i;

		elenco = leggiDich("dichiarazioni.txt");
		vett = filtra(elenco, "Renz", &dim);
		ordina(vett, dim);
		for (i= 0; i<dim; i++) {
			printf("%d/%d/%d %s %s\n",
				vett[i].data.giorno,
				vett[i].data.mese,
				vett[i].data.anno,
				vett[i].nome,
				vett[i].testo);
		}
		freelist(elenco);
		free(vett);
	}
	{ // Es. 3 && 4
		list elenco;
		char keyword[4096];
		char * nome;
		elenco = leggiDich("dichiarazioni.txt");
		printf("Inserire al keyword (senza spazi): ");
		scanf("%s", keyword);
		nome = commenti(elenco, keyword);
		if (nome != NULL) {
			printf("La persona che ha fatto più commenti sull'argomento %s e'
%s\n",
				keyword,
				nome);
			free(nome);
		}
		else{
			printf("Non esistono commenti sull'argomento %s\n", keyword);
		}
		freelist(elenco);
	}
	return 0;
```

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

Prova d'Esame 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

}

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

Prova d'Esame 6A di Giovedì 8 Settembre 2016 – tempo a disposizione 2h

"dichiarazioni.txt":

```
01/01/1970 BertandRussel ask_yourself_only_what_are_the_facts
06/09/2016 MatteoRenzi Sul_sisma_no_a_battaglie_politiche
04/08/2016 MatteoRenzi Italia_in_pole,_ma_ora_basta_polemiche
06/09/2016 MarcoDiVaio Ritrovarsi_dopo_Torino,_convinti_di_aver_migliorato_la_rosa
06/09/2016 RenzoUlivieri Il_Bologna_puo'_ripercorrere_le_orme_del_Toro
```