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

Prova d'Esame 5A di Giovedì 7 Luglio 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 noto quotidiano online pubblica nel proprio sito web (nella colonna più a destra) video di intrattenimento generico, con lo scopo di attirare più lettori possibili per più tempo possibile. A tal scopo, ha elaborato un raffinatissimo sistema tramite il quale tiene traccia di ogni volta che un video viene visto da un utente.
In particolare, in un file di testo, vengono memorizzate le informazioni relative ad ogni evento di visualizzazione, ogni evento su una riga distinta. Per ogni evento di visualizzazione, vengono memorizzati: l'**identificatore unico del video** (un intero); l'**identificatore unico dell'utente** (un intero); il **giorno** dell'anno in cui il video è stato visto (un intero); la durata in secondi della visione (un intero); e infine una **descrizione** in linguaggio naturale del video stesso (una stringa di al più 1023 caratteri utili, contenente spazi). Ognuno dei campi sopra descritti è separato da uno spazio; ogni riga, compresa l'ultima, è sempre terminata da un carattere "newline" ('\n').
Si vedano, a titolo di esempio, il file di testo "video.txt" fornito nello StartKit.

Esercizio 1 – Struttura dati Video, e funzioni di lett./scritt. (mod. element.h e video.h/.c)

Si definisca un'opportuna struttura dati **`Video`**, al fine di rappresentare l'evento di visione di un video da parte di un utente (id del video stesso, id utente, giorno, durata, e descrizione).

Si definisca la funzione:

```
Video leggiUnVideo(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati **`FILE`**, legga da tale file le informazioni relative ad un solo video, e le restituisca in forma di una struttura dati di tipo **`Video`**. Qualora la funzione incontri dei problemi nella lettura (ad esempio perché si è giunti al termine del file), la funzione dovrà restituire comunque una struttura dati di tipo Video, ma dovrà porre tutti i campi di tipo intero al valore -1, e il campo descrizione a stringa vuota ("").

Si definisca la funzione:

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

che, ricevuta in ingresso una stringa rappresentante il nome di un file contenente le informazioni sulle visioni dei video, legga da tale file tali informazioni e li tramite una lista di strutture dati di tipo **`Video`**. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve stampare un messaggio di errore a video, e poi deve restituire una lista vuota. A tal scopo, la funzione usi la funzione leggiUnVideo(…) definita precedentemente.

Si definisca la procedura:

```
void stampaVideo(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Video`**, stampi a video le informazioni riguardo le visualizzazioni dei video.

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 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

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

Per poter comprendere appieno quali sono i video più visti, è necessario filtrare via i video che sono stati visti meno di un certo numero di secondi. A tal scopo il candidato definisca una procedura:

```
Video * filtra(list elenco, int durataMinima, int * dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Video`** e la durata minima in secondi, restituisca in uscita un vettore contenente tutte le strutture dati di tipo Video, relative a visioni di lunghezza pari o maggiore alla durata minima specificata. Il vettore restituito dovrà essere stato allocato dinamicamente della dimensione minima necessaria, e tramite il parametro **`dim`** passato per riferimento, si dovrà restituire la dimensione di tale vettore.

E' necessario ordinare il vettore di strutture dati di tipo **`Video`** (ottenuto tramite la funzione di cui sopra) in base alla durata della visione, in ordine decrescente (prima i video che sono stati visti più a lungo). A parità di durata, si deve ordinare in base all'id del video, in ordine decrescente (prima i video con id maggiore). Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Video`** e la dimensione di tale vettore, ordini il vettore secondo i criteri specificati. A tal scopo, il candidato utilizzi l'algoritmo "Merge 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, 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)

Si vuole realizzare una funzionalità che permetta di conoscere i video che contengono nella descrizione una certa parola. Si definisca dapprima una funzione:

```
int presente(char * desc, char * keyword);
```

che ricevute in ingresso due stringhe **`desc`** e **`keyword`**, restituisca un valore interpretabile come vero se il parametro specificato **`keyword`** è contenuto nella stringa **`desc`**. A tal scopo, non si possono usare funzioni di libreria "string.h".

Il candidato definisca la funzione:

```
int freq(Video * v, int dim, char * keyword);
```

che, ricevuti come parametri un vettore di strutture dati di tipo **`Video`**, e la sua dimensione **`dim`**, e una stringa rappresentante una parola chiave, restituisca il numero di video presenti nel vettore, che contengono nella descrizione la keyword fornita in ingresso.

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, usando le informazioni fornite tramite i file di esempio fornito nello StartKit e le funzioni definite agli esercizi precedenti, chieda all'utente di specificare una parola chiave, e stampi quanti sono i video che hanno durata maggiore di 100 secondi, e che contengono nella descrizione la parola 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 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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

#define DIM 1024



typedef struct {
	int idVideo;
	int idUtente;
	int giorno;
	int durata;
	char desc[DIM];
} Video;




typedef Video element;

int compare(Video v1, Video v2);


#endif
```

"element.c":

```
#include "element.h"

int compare(Video v1, Video v2) {
	int result;
	result = v2.durata - v1.durata;
	if (result == 0)
		result = v2.idVideo - v1.idVideo;
	return result;
}
```

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

Prova d'Esame 5A di Giovedì 7 Luglio 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 */
```

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

Prova d'Esame 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

```
{
  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 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

"video.h":

```
#ifndef _VIDEO_H
#define _VIDEO_H

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

#include <stdlib.h>

// Es. 1
Video leggiUnVideo(FILE * fp);
list leggiTuttiVideo(char * fileName);
void stampaVideo(list elenco);

// Es. 2
Video * filtra(list elenco, int durataMinima, int * dim);
void ordina(Video * v, int dim);

// Es. 3
int presente(char * desc, char * keyword);
int freq(Video * v, int dim, char * keyword);


#endif
```

"video.c":

```
#include "video.h"

Video leggiUnVideo(FILE * fp) {
	Video result;
	Video err;
	int i;
	int errore = 0;
	char ch;

	err.idVideo = -1;
	err.idUtente = -1;
	err.giorno = -1;
	err.durata = -1;
	err.desc[0] = '\0';

	if (fscanf(fp, "%d%d%d%d",
		&(result.idVideo), &(result.idUtente), &(result.giorno), &(result.durata))
== 4) {
		fgetc(fp); // tolgo lo spazio di separazione
		i=0;
		do {
			ch=fgetc(fp);
			if (ch!='\n' && i<DIM-1) {
				result.desc[i] = ch;
				i++;
			}
		} while (ch!='\n' && i<DIM-1);
		if (ch!='\n')
			errore = 1;
```

```
		else
			result.desc[i] = '\0';
	}
	else
		errore = 1;

	if (errore)
		return err;
	else
		return result;
}

list leggiTuttiVideo(char * fileName) {
	list result;
	FILE * fp;
	Video temp;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		do {
			temp = leggiUnVideo(fp);
			if (temp.idVideo!=-1)
				result = cons(temp, result);
		} while (temp.idVideo!=-1);
		fclose(fp);
	}
	else {
		printf("Errore di apertura file: %s\n", fileName);
	}
	return result;
}

void stampaVideo(list elenco) {
	Video temp;
	while (!empty(elenco)) {
		temp = head(elenco);
		printf("%d %d %d %d %s\n", temp.idVideo, temp.idUtente, temp.giorno,
temp.durata, temp.desc);
		elenco = tail(elenco);
	}
	return;
}

Video * filtra(list elenco, int durataMinima, int * dim) {
	Video * result = NULL;
	list temp;
	int i;

	*dim = 0;
	temp = elenco;
	while (!empty(temp)) {
		if (head(temp).durata >= durataMinima)
			*dim = *dim + 1;
		temp = tail(temp);
	}
```

```
	result = (Video *) malloc(sizeof(Video) * *dim);
	i = 0;
	while (!empty(elenco)) {
		if (head(elenco).durata >= durataMinima) {
			result[i] = head(elenco);
			i++;
		}
		elenco = tail(elenco);
	}
	return result;
}

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

int presente(char * desc, char * keyword) {
	int i;
	int j;
	int k;
	int trovato;

	trovato = 0;
	i = 0;
	j = 0;
	k = 0;
	while (i<strlen(desc) && j<strlen(keyword) && !trovato) {
		if (desc[i] == keyword[j]) {
			k= i+1;
			j++;
```

```
                trovato = 1;
                while(desc[k] == keyword[j] && trovato) {
                        k++; j++;
                }
                if (keyword[j] != '\0') {
                        trovato = 0;
                        j = 0;
                }
        }
        i++;
    }
    return trovato;
}

int freq(Video * v, int dim, char * keyword) {
    int result;
    int i;

    result = 0;
    for (i=0; i<dim; i++) {
        if (presente(v[i].desc, keyword))
            result++;
    }
    return result;
}
```

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

Prova d'Esame 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{ // Es. 1
		list elenco;
		elenco = leggiTuttiVideo("video.txt");
		stampaVideo(elenco);
		freelist(elenco);
	}
	{ // Es. 2
		list elenco;
		Video * filtrato;
		int dim;
		int i;
		elenco = leggiTuttiVideo("video.txt");
		filtrato = filtra(elenco, 100, &dim);
		printf("\n\n");
		for (i=0; i<dim; i++)
			printf("%d %d %d %d %s\n", filtrato[i].idVideo, filtrato[i].idUtente,
filtrato[i].giorno, filtrato[i].durata, filtrato[i].desc);
		ordina(filtrato, dim);
		printf("\n\n");
		for (i=0; i<dim; i++)
			printf("%d %d %d %d %s\n", filtrato[i].idVideo, filtrato[i].idUtente,
filtrato[i].giorno, filtrato[i].durata, filtrato[i].desc);
		freelist(elenco);
		free(filtrato);
	}
	{ // Es. 3 && 4
		list elenco;
		Video * filtrato;
		int dim;
		char keyword[DIM];

		elenco = leggiTuttiVideo("video.txt");
		filtrato = filtra(elenco, 100, &dim);
		ordina(filtrato, dim);
		printf("Che parola si vuole cercare? ");
		scanf("%s", keyword);
		printf("Ricerca della parola %s, la contengono: %d video\n", keyword,
freq(filtrato, dim, keyword));

		freelist(elenco);
		free(filtrato);
	}
	return 0;
}
```

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

Prova d'Esame 5A di Giovedì 7 Luglio 2016 – tempo a disposizione 2h

"video.txt":

```
19 234567 187 180 Sasa' calciatore nazionale ho sbagliato il rigore ma ho dato tutto
12 234567 188 230 Fessi' noto calciatore condannato per evasione
12 345678 188 220 Fessi' noto calciatore condannato per evasione
17 456789 188 15 Lano Binfi noto attore dice mi manca solo un premio
17 234567 188 7 Lano Binfi noto attore dice mi manca solo un premio
12 123456 188 11 Fessi' noto calciatore condannato per evasione
19 123456 189 130 Sasa' calciatore nazionale ho sbagliato il rigore ma ho dato tutto
19 234567 189 137 Sasa' calciatore nazionale ho sbagliato il rigore ma ho dato tutto
```