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

Prova d'Esame 4A di Giovedì 16 Giugno 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"*.

Una nuova ed innovativa azienda informatica ha creato una piattaforma per relazioni "social", dove gli utenti possono pubblicare dei brevi messaggi. Tali brevi messaggi vengono chiamati **Post**, e vengono memorizzati su di un file di testo, al fine di permettere all'azienda di compiere alcune analisi di tali post.
In particolare, in tale file viene memorizzato un Post su ogni riga: all'inizio un **identificatore unico del post** stesso (un intero); a seguire, separato da uno spazio, un **identificatore unico dell'utente** (un intero); separata da uno spazio, la **data** e l'**ora** del posto, nel formato "aaaa/mm/gg hh:mm"; infine, dopo uno spazio, il messaggio del post (una stringa di al più 255 caratteri utili, contenente anche spazi e altri caratteri di punteggiatura, ma non caratteri 'newline'). Ogni riga (compresa l'ultima) è terminata da un carattere 'newline' ('\n'). Non è noto a priori quanti post siano memorizzati nel file.
Si vedano, a titolo di esempio, il file di testo "social.txt" fornito nello StartKit.

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

Si definisca un'opportuna struttura dati **`Post`**, al fine di rappresentare un post (id del post stesso, id utente, data, ora, e messaggio). A tal scopo, si definiscano anche due opportune strutture dati **`Data`** e **`Ora`**, al fine di rappresentare una data specifica (anno, mese e giorno) ed un orario specifico (ora e minuto).

Si definisca la funzione:

```
int leggiUnPost(FILE * fp, Post * dest);
```

che, ricevuto in ingresso un puntatore ad una struttura dati **`FILE`**, legga da tale file le informazioni relative ad un solo post, e le restituisca tramite il parametro **`dest`** passato "per riferimento". La funzione dovrà restituire un valore intero, interpretabile come "vero" se è stato possibile leggere un post, come "falso" in caso contrario.

Si definisca la funzione:

```
Post * leggiTuttiPost(char * fileName, int * dim);
```

che, ricevuta in ingresso una stringa rappresentante il nome di un file contenente dei post, legga da tale file i post e li salvi in un vettore allocato dinamicamente. Tramite il parametro **`dim`**, passato per riferimento, la funzione deve restituire la dimensione del vettore. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve stampare un messaggio di errore a video, e poi deve restituire un puntatore a **`NULL`** e come valore della dimensione il valore zero. A tal scopo, la funzione usi la funzione leggiUnPost(…) definita precedentemente.

Si definisca la procedura:

```
void stampaPost(Post * v, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Post`**, e la sua dimensione **`dim`**, stampi a video i post memorizzati nel vettore passato come argomento.

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 4A di Giovedì 16 Giugno 2016 – tempo a disposizione 2h

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

Il candidato infine definisca una procedura:

```
void ordinaPost(Post * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Post`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: innanzitutto in base alla data, in ordine crescente; a parità di data, in base all'orario, in ordine crescente; a parità di orario, in base all'identificatore del post, ancora in ordine crescente. A tal fine, il candidato utilizzi l'algoritmo "quick sort" visto a lezione.

Accade poi che degli utenti pubblichino dei post con testo vuoto. Tali post ovviamente non devono essere considerati. Il candidato definisca una funzione:

```
list cancellaVuoti(Post * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Post`** e la dimensione di tale vettore, restituisca una lista contenente i soli post con messaggio non vuoto. La lista deve essere ordinata, secondo lo stesso criterio "data, ora, id del post" di cui alla funzione precedente.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Esercizio 3 – Identificazione dell'utente che effettua più post (modulo social.h/social.c)

Si vuole realizzare una funzionalità che permetta alla piattaforma di determinare l'utente che ha pubblicato più post in assoluto.
Si definisca dapprima una funzione:

```
int * utenti(list p, int * dim);
```

che ricevuta la lista contenente i post, restituisca in uscita un vettore di interi, allocato dinamicamente (non necessariamente della dimensione minima possibile), contenente gli identificatori degli utenti che hanno effettuato dei post. Tale vettore ovviamente non dovrà contenere ripetizioni. Tramite il parametro **`dim`**, passato per riferimento, si dovrà restituire la dimensione del vettore.

Il candidato definisca la funzione:

```
int best(list p, int * utenti, int dim);
```

che, ricevuti come parametri una lista di strutture dati di tipo **`Post`**, e un vettore di interi rappresentante gli identificatori degli utenti, e la dimensione **`dim`** di tale vettore, restituisca l'identificatore dell'utente che ha fatto più post.

Esercizio 4 – Stampa degli utenti che scrivono post, stampa dell'utente che effettua più post di tutti, 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 degli utenti (senza ripetizioni), e stampi a video l'identificatore dell'utente che ha effettuato più post di tutti.

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 4A di Giovedì 16 Giugno 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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

#define DIM 256

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

typedef struct {
	int ora;
	int minuto;

} Ora;

typedef struct {
	int idPost;
	int idUtente;
	Data data;
	Ora ora;
	char mesg[DIM];
} Post;

typedef Post element;

int compare(Post p1, Post p2);
#endif
```

"element.c":

```
#include "element.h"

int compare(Post p1, Post p2) {
	int result;
	result = p1.data.anno - p2.data.anno;
	if (!result)
		result = p1.data.mese - p2.data.mese;
	if (!result)
		result = p1.data.giorno - p2.data.giorno;

	if (!result)
		result = p1.ora.ora - p2.ora.ora;
	if (!result)
		result = p1.ora.minuto - p2.ora.minuto;

	if (!result)
		result = p1.idPost - p2.idPost;
	return result;
}
```

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

Prova d'Esame 4A di Giovedì 16 Giugno 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 showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%d %d %d/%d/%d %d:%d %s\n",
                     temp.idPost, temp.idUtente,
                     temp.data.anno, temp.data.mese, temp.data.giorno,
                     temp.ora.ora, temp.ora.minuto,
                     temp.mesg);
              return showlist(tail(l));
       }
       else {
              printf("\n\n");
              return;
       }
}
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 (compare(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, 2015/2016 – Modulo 2

Prova d'Esame 4A di Giovedì 16 Giugno 2016 – tempo a disposizione 2h

"social.h":

```
#ifndef _SOCIAL_H
#define _SOCIAL_H

#include "stdio.h"
#include "stdlib.h"
#include "string.h"

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

// es. 1
int leggiUnPost(FILE * fp, Post * dest);
Post * leggiTuttiPost(char * fileName, int * dim);
void stampaPost(Post * v, int dim);


// Es. 2
void ordinaPost(Post * v, int dim);
list cancellaVuoti(Post * v, int dim);


// Es. 3
int * utenti(list p, int * dim);
int best(list p, int * utenti, int dim);

#endif
```

"social.c":

```
#include "social.h"

int leggiUnPost(FILE * fp, Post * dest) {
	int result = 0;
	int ok;
	char ch;
	int i;

	ok = fscanf(fp, "%d %d %d/%d/%d %d:%d",
		&(dest->idPost), &(dest->idUtente),
		&(dest->data.anno), &(dest->data.mese), &(dest->data.giorno),
		&(dest->ora.ora), &(dest->ora.minuto) );

	// butto via lo spazio tra l'ora e il messaggio
	if (ok == 7)
		ok = (fgetc(fp) == ' ');

	if ( ok ) {
		i = 0;
		while ((ch=fgetc(fp))!= '\n' && i<DIM-1) {
			dest->mesg[i] = ch;
			i++;
		}
		dest->mesg[i] = '\0';
		if (ch == '\n')
```

```
                result = 1;
        }
        return result;
}


Post * leggiTuttiPost(char * fileName, int * dim) {
        FILE * fp;
        Post * result = NULL;
        Post temp;
        int cont;

        *dim = 0;
        fp = fopen(fileName, "rt");
        if (fp != NULL) {
                // ciclo di conta di quanti elementi ci sono nel file
                cont = 0;
                while (leggiUnPost(fp, &temp))
                        cont++;

                // alloco e "riavvolgo" il file
                rewind(fp);
                result = (Post*) malloc(sizeof(Post) * cont);

                // lettura vera e propria
                while (leggiUnPost(fp, &temp)) {
                        result[*dim] = temp;
                        *dim = *dim + 1;
                }

                fclose(fp);
        }
        else {
                printf("Problema nell'apertura del file: %s\n", fileName);
        }
        return result;
}



void stampaPost(Post * v, int dim) {
        int i;
        for (i=0; i<dim; i++) {
                printf("%d %d %d/%d/%d %d:%d %s\n",
                        v[i].idPost, v[i].idUtente,
                        v[i].data.anno, v[i].data.mese, v[i].data.giorno,
                        v[i].ora.ora, v[i].ora.minuto,
                        v[i].mesg);
        }
        printf("\n\n");
}


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

```
}

void quickSortR(Post a[], int iniz, int fine)
{
	int i, j, iPivot;
	Post pivot;
	if (iniz < fine) {
		i = iniz;
		j = fine;
		iPivot = fine;
		pivot = a[iPivot];
		do  {/* trova la posizione del pivot */
			while (i < j && compare(a[i], pivot) <= 0) i++;
			while (j > i && compare(a[j], pivot) >= 0) j--;
			if (i < j) scambia(&a[i], &a[j]);
		}
		while (i < j);

		if (i != iPivot && compare(a[i], a[iPivot]) != 0) {
			scambia(&a[i], &a[iPivot]);
			iPivot = i;
		}
		/* ricorsione sulle sottoparti, se necessario */
		if (iniz < iPivot - 1)
			quickSortR(a, iniz, iPivot - 1);
		if (iPivot + 1 < fine)
			quickSortR(a, iPivot + 1, fine);
	} /* (iniz < fine) */
} /* quickSortR */



void quickSort(Post a[], int dim)
{
	quickSortR(a, 0, dim - 1);
}

void ordinaPost(Post * v, int dim) {
	quickSort(v, dim);
}


list cancellaVuoti(Post * v, int dim) {
	list result;
	int i;

	result = emptylist();

	for (i=0; i<dim; i++) {
		if (strlen(v[i].mesg)>0)
			result = insord_p(v[i], result);
	}
	return result;
}


int member(int * v, int dim, int el) {
```

```
	int i;
	int trovato = 0;
	for (i=0; i<dim && !trovato; i++)
		if (v[i] == el)
			trovato = 1;
	return trovato;
}

int * utenti(list p, int * dim) {
	list temp;
	int cont;
	int * result;

	cont = 0;
	temp = p;
	while (!empty(temp)) {
		cont++;
		temp = tail(temp);
	}

	result = (int*) malloc(sizeof(int) * cont);
	*dim = 0;
	while (!empty(p)) {
		if (!member(result, *dim, head(p).idUtente)) {
			result[*dim] = head(p).idUtente;
			*dim = *dim + 1;
		}
		p = tail(p);
	}
	return result;
}

int conta(list p, int utente) {
	int result = 0;
	while (!empty(p)) {
		if (head(p).idUtente == utente)
			result++;
		p = tail(p);
	}
	return result;
}

int best(list p, int * utenti, int dim) {
	int i;
	int max;
	int result;

	max = -1;
	result = -1;
	for (i=0; i<dim; i++) {
		if (conta(p, utenti[i]) > max) {
			max = conta(p, utenti[i]);
			result = utenti[i];
		}
	}
	return result;
}
```

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

Prova d'Esame 4A di Giovedì 16 Giugno 2016 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{ // Es. 1
		Post * v;
		int dim;
		v = leggiTuttiPost("social.txt", &dim);
		stampaPost(v, dim);
		free(v);
	}
	{ // Es. 2
		Post * v;
		int dim;
		list elenco;

		v = leggiTuttiPost("social.txt", &dim);
		ordinaPost(v, dim);
		stampaPost(v, dim);
		elenco = cancellaVuoti(v, dim);
		showlist(elenco);
		free(v);
		freelist(elenco);
	}
	{ // Es. 3 && 4
		Post * v;
		int dim;
		list elenco;
		int * ut;
		int numUtenti;
		int i;
		int theBest;

		v = leggiTuttiPost("social.txt", &dim);
		elenco = cancellaVuoti(v, dim);
		ut = utenti(elenco, &numUtenti);
		for (i=0; i<numUtenti; i++)
			printf("%d\t", ut[i]);
		printf("\n\n");

		theBest = best(elenco, ut, numUtenti);
		printf("L'utente che posta di piu' e': %d\n", theBest);

		free(v);
		free(ut);
		freelist(elenco);
	}
	return;
}
```

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

Prova d'Esame 4A di Giovedì 16 Giugno 2016 – tempo a disposizione 2h

"social.txt":

```
1256816 34 2016/06/15 10:42 Secondo me non ha giocato tanto bene...troppo catenaccio...
1456794 12 2016/06/15 10:38 Hai visto che paritta ha fatto l'Italia?
1456803 27 2016/06/15 10:38 Certo che ho visto la partita!!!
1256842 34 2016/06/15 11:24 Secondo me non ha giocato tanto bene...troppo catenaccio...
1456829 27 2016/06/15 10:58 Boh, ho visto la partita... speriamo faccia uguale alla prossima
1256811 34 2016/06/15 10:41
1256815 34 2016/06/15 10:42 Scusate ho inviato per sbaglio con testo vuoto
1456827 12 2016/06/15 10:49 Ma cosa dici? e' stata una partita bellissima! Forza Italia!!!
```