Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – 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"*.

In un grande stato europeo si tiene ogni anno una corsa ciclistica a tappe, dove per ogni tappa viene cronometrato il tempo di percorrenza di ogni atleta ciclista. Vince la corsa l'atleta con la somma minore dei tempi.
In particolare su un file di testo, per ogni tappa e per ogni ciclista, vengono memorizzate le seguenti informazioni: nome della **tappa** (una stringa di al più 23 caratteri utili, senza spazi), il nome del **ciclista** (una stringa di al più 63 caratteri utili, senza spazi), e il **tempo** impiegato per completare la tappa, nel formato hh:mm:ss (tre interi). Se un ciclista non termina la tappa o non si presenta, il sistema scrive sul file un tempo negativo.

Esercizio 1 – Strutture dati Tempo, Tappa, e funzioni di lett./scritt. (mod. element.h/.c e tappe.h/.c)

Si definisca un'opportuna struttura dati **`Tempo`** nel file "element.h", atta a rappresentare una durata temporale in termini di ore, minuti e secondi. Si definisca poi una opportuna struttura dati **`Tappa`**, al fine di rappresentare i dati relativi ad una singola tappa, cioè il nome della tappa, il nome del ciclista, e il tempo impiegato.

Si definiscano nel modulo element.h/.c le seguenti fuzioni:

```
int compareTempo(Tempo t1, Tempo t2);
Tempo somma(Tempo t1, Tempo t2);
```

La **`compare(…)`** confronta due tempi, e restituisce un valore negativo, pari a zero o positivo a seconda che **`t1`** sia inferiore, pari o superiore a **`t2`**. La **`somma(…)`** invece esegue la somma di due tempi e restituisce una nuova struttura dati di tipo **`Tempo`**.

Si definisca nel modulo tappe.h/.c la funzione:

```
list leggiTappe(char * nomeFile);
```

che, ricevuto in ingresso il nome di un file contenente i dati relativi alle tappe, ai ciclisti e ai tempi di percorrenza, legga da tale file le informazioni e le restituisca tramite una lista di strutture dati di tipo **`Tappa`**. In caso di errore nell'apertura del file, la funzione deve restituire una lista vuota. Qualora il tempo risulti essere negativo, si deve salvare come tempo di tappa un valore pari a 99:59:59.

Si definisca la procedura:

```
void showList(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Tappa`** stampi a video il contenuto di tali strutture dati.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra. Il candidato abbia cura anche di deallocare eventuale memoria allocata dinamicamente nei test.

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – tempo a disposizione 2h

Esercizio 2 – Tempi del singolo ciclista (modulo tappe.h/.c)

Il candidato definisca una funzione:

```
Tappa * singoloCiclista(list elenco, char * nome, int * dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Tappa`**, e il nome di un ciclista, restituisca un vettore di strutture dati di tipo **`Tappa`**, allocato dinamicamente della dimensione minima necessaria, contenente tutte e sole le informazioni relative al ciclista specificato tramite il parametro **`nome`**. Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore allocato dinamicamente. Il vettore restituito deve essere ordinato in senso lessicografico sulla base del nome della tappa. A tal fine, il candidato utilizzi uno qualunque degli algoritmi di ordinamento 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. In particolare, il candidato stampi a video i dati relativi alle tappe di un ciclista a scelta tra quelli presenti nel file fornito nello StartKit. Il candidato abbia cura anche di deallocare eventuale memoria allocata dinamicamente nei test.

Il candidato definisca una funzione:

```
Tappa sommaCiclista(Tappa * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Tappa`** tutte relative allo stesso ciclista, e la dimensione del vettore **`dim`**, restituisca una nuova struttura dati di tipo **`Tappa`**, dove il nome del ciclista è sempre lo stesso, il nome della tappa è **`"Totale"`**, e il tempo è pari alla SOMMA di tutti tempi memorizzati nel vettore d'ingresso.

Esercizio 3 – Stampa del vincitore (modulo tappe.h/tappe.c)

Si definisca una procedura:

```
void vincitore(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Tappa`**, stampi a video il nome del ciclista che ha impiegato meno tempo totale a percorrere tutte le tappe, assieme al tempo totale impiegato.

Esercizio 4 – Stampa del vincitore, 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 il nome ed il tempo totale del vincitore.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste, se non in structure sharing (in tal caso, si segnali la situazione con un breve commento nel codice).

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – tempo a disposizione 2h

"element.h":

```
#include <stdio.h>
#include <string.h>
#ifndef _ELEMENT_H
#define _ELEMENT_H

typedef struct {
	int ore;
	int minuti;
	int secondi;
} Tempo;
typedef struct {
	char tappa[24];
	char ciclista[64];
	Tempo tempo;
} Tappa;
typedef Tappa element;

int compareTempo(Tempo t1, Tempo t2);
Tempo somma(Tempo t1, Tempo t2);
int compareTappa(Tappa t1, Tappa t2);
#endif
```

"element.c":

```
#include "element.h"

int compareTempo(Tempo t1, Tempo t2) {
	int result;

	result = t1.ore-t2.ore;
	if (result == 0)
		result = t1.minuti - t2.minuti;
	if (result == 0)
		result = t1.secondi - t2.secondi;

	return result;
}

Tempo somma(Tempo t1, Tempo t2) {
	Tempo result;
	result.secondi = t1.secondi+t2.secondi;
	result.minuti = result.secondi / 60;
	result.secondi = result.secondi % 60;

	result.minuti = result.minuti + t1.minuti + t2.minuti;
	result.ore = result.minuti / 60;
	result.minuti = result.minuti % 60;

	result.ore = result.ore + t1.ore + t2.ore;

	return result;
}

int compareTappa(Tappa t1, Tappa t2) {
	return strcmp(t1.tappa, t2.tappa);
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – 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 %d:%d:%d\n", temp.tappa, temp.ciclista,
                     temp.tempo.ore, temp.tempo.minuti, temp.tempo.secondi);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}


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

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – tempo a disposizione 2h

"tappe.h":

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


#ifndef _TAPPE_H
#define _TAPPE_H

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

list leggiTappe(char * nomeFile);

Tappa * singoloCiclista(list elenco, char * nome, int * dim);
Tappa sommaCiclista(Tappa * v, int dim);

void vincitore(list elenco);

#endif
```

"tappe.c":

```
#include "tappe.h"

list leggiTappe(char * nomeFile) {
	FILE * fp;
	list result;
	Tappa temp;
	Tempo zero;

	result = emptylist();
	fp = fopen(nomeFile, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura file %s\n", nomeFile);
		return result;
	}
	else {
		zero.ore = 0;
		zero.minuti = 0;
		zero.secondi = 0;
		while (fscanf(fp, "%s %s %d:%d:%d",
			temp.tappa, temp.ciclista, &(temp.tempo.ore), &(temp.tempo.minuti),
&(temp.tempo.secondi)) == 5)
		{
			if (compareTempo(temp.tempo, zero) <= 0) {
				temp.tempo.ore = 99;
				temp.tempo.minuti = 59;
				temp.tempo.secondi = 59;
			}
			result = cons(temp, result);
		}
		fclose(fp);
		return result;
```

```
	}
}


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

void bubbleSort(Tappa v[], int n) {
	int i, ordinato = 0;
	while (n>1 && !ordinato) {
		ordinato = 1;
		for (i=0; i<n-1; i++)
			if (compareTappa(v[i], v[i+1])>0) {
				scambia(&v[i],&v[i+1]);
				ordinato = 0;
			}
			n--;
	}
}

Tappa * singoloCiclista(list elenco, char * nome, int * dim) {
	Tappa * result;
	list temp;

	temp = elenco;
	*dim = 0;
	while (!empty(temp)) {
		if (strcmp(head(temp).ciclista, nome)==0)
			*dim = *dim+1;
		temp = tail(temp);
	}

	temp = elenco;
	result = (Tappa *) malloc(sizeof(Tappa) * *dim);
	*dim = 0;

	while (!empty(temp)) {
		if (strcmp(head(temp).ciclista, nome)==0) {
			result[*dim] = head(temp);
			*dim = *dim+1;
		}
		temp = tail(temp);
	}
	bubbleSort(result, *dim);
	return result;
}


Tappa sommaCiclista(Tappa * v, int dim) {
	int i;
	Tappa result;

	if (dim>0) {
		strcpy(result.ciclista, v[0].ciclista);
```

```
            strcpy(result.tappa, "Totale");
            result.tempo = v[0].tempo;
            for (i=1; i<dim; i++) {
                  result.tempo = somma(result.tempo, v[i].tempo);
            }
      }
      return result;
}


void vincitore(list elenco) {
      list temp;
      Tappa t;
      Tappa record;
      Tappa * v;
      int dim;

      temp = elenco;
      if (!empty(temp)) {
            v = singoloCiclista(elenco, head(temp).ciclista, &dim);
            record = sommaCiclista(v, dim);
            free(v);
            temp = tail(temp);
      }
      else {
            strcpy(record.tappa, "ERRORE");
            strcpy(record.ciclista, "ERRORE");
      }

      while(!empty(temp)) {
            v = singoloCiclista(elenco, head(temp).ciclista, &dim);
            t = sommaCiclista(v, dim);
            if (compareTempo(t.tempo,record.tempo)<0)
                  record = t;
            free(v);
            temp = tail(temp);
      }
      printf("%s %s %d:%d:%d\n", record.tappa, record.ciclista,
                  record.tempo.ore, record.tempo.minuti, record.tempo.secondi);
      return;
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>

#include "tappe.h"

int main() {

	{	// ESERCIZIO 1
		list elenco;
		elenco = leggiTappe("corsa.txt");
		showlist(elenco);
		freelist(elenco);
	}

	{	// ESERCIZIO 2
		list elenco;
		Tappa * ciclista;
		Tappa totale;
		int i, dim;

		elenco = leggiTappe("corsa.txt");
		ciclista = singoloCiclista(elenco, "FedericoChesani", &dim);
		for (i=0; i<dim; i++) {
			printf("%s %s %d:%d:%d\n", ciclista[i].tappa, ciclista[i].ciclista,
			ciclista[i].tempo.ore, ciclista[i].tempo.minuti,
ciclista[i].tempo.secondi);
		}
		printf("\n\n");
		totale = sommaCiclista(ciclista, dim);
		printf("%s %s %d:%d:%d\n", totale.tappa, totale.ciclista,
			totale.tempo.ore, totale.tempo.minuti, totale.tempo.secondi);
		freelist(elenco);
		free(ciclista);
	}

	{	// ESERCIZIO 3
		list elenco;

		elenco = leggiTappe("corsa.txt");
		printf("VINCITORE:\n");
		vincitore(elenco);
		printf("\n\n");
		freelist(elenco);
	}


	return 0;
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 5A di Giovedì 9 Luglio 2015 – tempo a disposizione 2h

"corsa.txt":

```
tappa01 FedericoChesani 3:13:48
tappa01 CarloGiannelli 3:12:01
tapap01 VincenzoNibali 2:34:23
tappa02 CarloGiannelli 1:15:00
tappa02 FedericoChesani 1:16:21
tapap02 VincenzoNibali 0:58:41
tappa03 FedericoChesani -1:00:00
tappa03 CarloGiannelli 5:58:58
tapap03 VincenzoNibali 4:59:59
tappa04 CarloGiannelli 2:37:46
tapap04 VincenzoNibali 2:31:28
tappa04 FedericoChesani -1:00:00
```