Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – 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 società sportiva di ciclismo organizza ogni anno una gara riservata a squadre, dove non vince il singolo ciclista, ma la squadra i cui componenti percorrono in totale il maggior numero di chilometri. In particolare, la gara prevede tre percorsi detti "corto" (50 Km), "medio" (105 Km) e "lungo" (157 Km). Per evitare inganni, lungo i percorsi (e prima dell'arrivo) vi sono tre controlli a sorpresa da parte dei giudici. Dopo un controllo alla partenza, detto "**`controlloStart`**", vi è il "**`controlloA`**", comune a tutti e tre i percorsi; poi il "**`controlloB`**", comune solo al percorso medio e al percorso lungo; infine il "**`controlloC`**", che viene effettuato solo nel percorso lungo. Infine, vi è un controllo all'arrivo, detto "**`controlloEnd`**".
A fine gara su un file di testo vengono segnate le seguenti informazioni: ogni corridore è indicato su una riga, a partire dal **nome** (stringa di al più 255 caratteri utili, con spazi); separato dal carattere ';', vi è il nome della **squadra** (stringa di al più 255 caratteri utili, con spazi); poi, dopo un carattere di separazione ';', vi sono da una a cinque stringhe "**`controlloStart`**", "**`controlloA`**" "**`controlloB`**" "**`controlloC`**" "**`controlloEnd`**", separate tra loro da spazi, che rappresentano appunto il fatto che il ciclista sia passato o meno sotto un certo controllo (si noti che "**`controlloStart`**" c'è sempre per tutti i ciclisti nel file). Siccome qualcuno imbroglia, non è detto che le stringhe vi siano sempre tutte: potrebbero essercene una, due, etc. Ad esempio, un ciclista potrebbe avere effettuato tutti i controlli intermedi, ma non essere arrivato alla fine: in tal caso, è come se non avesse partecipato. Ogni riga è sempre terminata da un carattere di fine linea '\n'.
Si veda, a titolo di esempio, il file di testo "gara.txt" fornito nello StartKit: non è noto a priori quanti dati siano memorizzati nei file.

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

Si definisca una opportuna struttura dati **`Gara`** per memorizzare i dati relativi ad un ciclista. In particolare, la struttura dovrà comprendere un campo per il nome del ciclista, un campo per la squadra, e cinque campi interi (da interpretare come booleani) per memorizzare il fatto che il ciclista sia passato o no in un controllo.
Si definisca la funzione:

```
Gara leggiUno(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo FILE, legga da tale file i dati relativi ad un solo ciclista, e li restituisca tramite una struttura dati di tipo **`Gara`**. Qualora il file sia terminato o non sia possibile leggere, la funzione deve restituire una struttura dati di tipo **`Gara`** il cui campo nome è pari alla stringa vuota "".
Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file contenente l'elenco dei ciclisti, legga da tale file i dati ivi contenuti, e li restituisca tramite una lista di strutture dati di tipo **`Gara`**. 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.
Si definisca infine la funzione:

```
int punti(Gara g);
```

che, ricevuta in ingresso una struttura dati di tipo gara, restituisca i punti guadagnati dal ciclista. Si assegna un punto per ogni km percorso (quindi 50 punti per il corto, 105 punti per il medio, 157 punti per il lungo). Si noti che per guadagnare i punti, un ciclista deve aver passato i controlli relativi al percorso fatto: ad esempio, per il percorso medio il ciclista deve avere i "**`controlloStart`**", "**`controlloA`**" "**`controlloB`**" "**`controlloEnd`**". Se manca un controllo, il punteggio restituito è pari a zero.
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, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – tempo a disposizione 2h

Esercizio 2 –Ordinamento e lettura delle squadre (moduli element.h/.c e gara.h/.c)

Si definisca una struttura dati **`Squadra`**, atta a memorizzare due informazioni: il nome di una squadra, e il totale dei punti guadagnati da quella squadra (un intero).

Il candidato definisca una procedura:

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

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Squadra`**, e la dimensione **`dim`** di tale vettore, ordini il vettore secondo il seguente criterio: in ordine decrescente rispetto al punteggio; a parità, in ordine alfabetico in base al nome della squadra. Il candidato utilizzi uno a scelta tra gli algoritmi di ordinamento visti a lezione.

Il candidato definisca una funzione:

```
Squadra * estrai(list elenco, int * dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Gara`**, e un intero **`dim`** passato per riferimento, restituisca un vettore allocato dinamicamente (della dimensione minima necessaria) di strutture dati di tipo **`Squadra`**. Il vettore risultante dovrà contenere l'elenco delle squadre i cui membri hanno partecipato alla gara. Il punteggio della squadra dovrà essere pari a zero. Ovviamente, il vettore restituito in uscita non dovrà contenere ripetizioni di squadre. Tramite il puntatore a intero **`dim`** la funzione dovrà restituire la dimensione del vettore allocato dinamicamente.

Esercizio 3 – Elaborazione classifica (modulo gara.h/gara.c)

La società organizzatrice della gara deve decidere chi è la squadra vincitrice. A tal scopo, assegna alla squadra i punteggi di ogni singolo membro di tale squadra: ad esempio, se per la squadra A hanno corso i ciclisti C1 (corto) e C2 (medio), il punteggio della squadra A sarà pari a 155.
A tal scopo si sviluppi una procedura:

```
void totale(Squadra * classifica, int dim, list elenco);
```

che, ricevuti in ingresso una lista di strutture dati di tipo **`Gara`**, un vettore di strutture dati di tipo **`Squadra`** e la dimensione di tale vettore, calcoli per ogni squadra il punteggio totale, e aggiorni in modo corrispettivo il vettore rappresentante la classifica.

Esercizio 4 – Stampa classifica totale, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che chieda all'utente di specificare il nome di un file contenente i risultati della gara, legga dai file tutte le informazioni e stampi a video la classifica delle squadre. La stampa deve avvenire in modo che le squadre con più punti (vincitrici) compaiano per prime. 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, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – tempo a disposizione 2h

"element.h":

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

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_NOME 255
#define DIM_SQUADRA 255

typedef struct {
	char nome[DIM_NOME];
	char squadra[DIM_SQUADRA];
	int controlloStart;
	int controlloA;
	int controlloB;
	int controlloC;
	int controlloEnd;
} Gara;

typedef Gara element;

typedef struct {
	char squadra[DIM_SQUADRA];
	int punti;
} Squadra;

int compare(Squadra s1, Squadra s2);

#endif
```

"element.c":

```
#include "element.h"

int compare(Squadra s1, Squadra s2) {
	if (s1.punti == s2.punti)
		return strcmp(s1.squadra, s2.squadra);
	else
		return s2.punti - s1.punti;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – 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 %d %d\n",
                     temp.nome, temp.squadra, temp.controlloStart, temp.controlloA,
temp.controlloB, temp.controlloC, temp.controlloEnd);
              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, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – tempo a disposizione 2h

"gara.h":

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

#ifndef _GARA_H
#define _GARA_H

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


// es. 1
Gara leggiUno(FILE * fp);
list leggi(char * fileName);
int punti(Gara g);

// es. 2
void ordina(Squadra * v, int dim);
Squadra * estrai(list elenco, int * dim);


// es. 3
void totale(Squadra * classifica, int dim, list elenco);

#endif
```

"gara.c":

```
#include "gara.h"

Gara leggiUno(FILE * fp) {
	Gara result;
	char ch;
	int i;
	char temp[15];

	strcpy(result.nome, "");
	i=0;
	while ((ch=fgetc(fp))!=';' && ch!='\n' && ch!=EOF && i<DIM_NOME-1) {
		result.nome[i++] = ch;
	}
	result.nome[i] = '\0';

	if (i>0) {
		i=0;
		while ((ch=fgetc(fp))!=';' && ch!='\n' && ch!=EOF && i<DIM_SQUADRA-1) {
		result.squadra[i++] = ch;
		}
		result.squadra[i] = '\0';

		result.controlloStart = 0;
		result.controlloA = 0;
		result.controlloB = 0;
		result.controlloC = 0;
		result.controlloEnd = 0;
```

```
            fscanf(fp, "%s", temp);
            if (strcmp("controlloStart", temp)==0)
                  result.controlloStart = 1;
            ch = fgetc(fp);
            while (ch==' ') {
                  fscanf(fp, "%s", temp);
                  if (strcmp("controlloA", temp)==0)
                        result.controlloA = 1;
                  if (strcmp("controlloB", temp)==0)
                        result.controlloB = 1;
                  if (strcmp("controlloC", temp)==0)
                        result.controlloC = 1;
                  if (strcmp("controlloEnd", temp)==0)
                        result.controlloEnd = 1;
                  ch = fgetc(fp);
            }
      }

      return result;
}


list leggi(char * fileName) {
      FILE * fp;
      list result;
      Gara temp;

      result = emptylist();
      fp = fopen(fileName, "rt");
      if (fp == NULL) {
            printf("Problema nell'apertura del file %s\n", fileName);
      }
      else {
            temp = leggiUno(fp);
            while (strcmp("", temp.nome) != 0) {
                  result = cons(temp, result);
                  temp = leggiUno(fp);
            }
            fclose(fp);
      }
      return result;
}

int punti(Gara g) {
      int result = 0;
      result = (g.controlloStart*g.controlloEnd) * (
            g.controlloA * 57 +
            g.controlloA * g.controlloB * (105-57) +
            g.controlloA * g.controlloB * g.controlloC * (157- 105 -57)
            );
      return result;
}


int squadraMember(char * squadra, list elenco) {
      if (empty(elenco))
```

```
                return 0;
        else {
                if (strcmp(squadra, head(elenco).squadra) == 0)
                        return 1;
                else
                        return squadraMember(squadra, tail(elenco));
        }
}

// Es. 2
void  merge(Squadra v[], int i1, int i2, int fine, Squadra 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(Squadra v[], int first, int last, Squadra 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(Squadra * v, int dim) {
        Squadra * temp = (Squadra *) malloc(sizeof(Squadra) * dim);
        mergeSort(v, 0, dim-1, temp);
        free(temp);
}



Squadra * estrai(list elenco, int * dim) {
        Squadra * result;
        Gara temp;
        list elencoTemp;
        int i;

        *dim = 0;
        result = NULL;
        elencoTemp = elenco;
        while (!empty(elencoTemp)) {
                temp = head(elencoTemp);
                if (!squadraMember(temp.squadra, tail(elencoTemp)))
                        *dim = *dim + 1;
                elencoTemp = tail(elencoTemp);
        }

        if (*dim > 0) {
```

```
            result = (Squadra *) malloc(sizeof(Squadra) * *dim);
            i = 0;
            while (!empty(elenco)) {
                  temp = head(elenco);
                  if (!squadraMember(temp.squadra, tail(elenco))) {
                        strcpy(result[i].squadra, temp.squadra);
                        result[i].punti = 0;
                        i++;
                  }
                  elenco = tail(elenco);
            }
      }
      return result;
}

void totale(Squadra * classifica, int dim, list elenco) {
      int i;
      list temp;
      Gara tempGara;

      for (i=0; i<dim; i++) {
            temp = elenco;
            classifica[i].punti = 0;
            while (!empty(temp)) {
                  tempGara = head(temp);
                  if (strcmp(classifica[i].squadra, tempGara.squadra)==0)
                        classifica[i].punti = classifica[i].punti + punti(tempGara);
                  temp = tail(temp);
            }
      }
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "gara.h"


int main() {

	{ // es. 1
		list elenco;
		elenco = leggi("gara.txt");
		showlist(elenco);
		printf("Punti del primo in lista: %d\n", punti(head(elenco)));
		freelist(elenco);
	}

	{ // es. 2
		list elenco;
		Squadra * classifica;
		int dim;
		int i;
		elenco = leggi("gara.txt");
		classifica = estrai(elenco, &dim);
		ordina(classifica, dim);
		for (i=0; i<dim; i++) {
			printf("%s %d\n", classifica[i].squadra, classifica[i].punti);
		}
		free(classifica);
		freelist(elenco);
	}

	{ // es. 3 && 4
		list elenco;
		Squadra * classifica;
		int dim;
		int i;
		elenco = leggi("gara.txt");
		classifica = estrai(elenco, &dim);
		totale(classifica, dim, elenco);
		ordina(classifica, dim);
		for (i=0; i<dim; i++) {
			printf("%s %d\n", classifica[i].squadra, classifica[i].punti);
		}
		free(classifica);
		freelist(elenco);
	}


	return 0;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 4A di Giovedì 14 Giugno 2018 – tempo a disposizione 2h

"gara.txt":

```
Federico Chesani;CUBO A.P.S.;controlloStart controlloA controlloB controlloC controlloEnd
Paola Mello;CUBO A.P.S.;controlloStart controlloA controlloB controlloC controlloEnd
Daniela Loreti;Borgo Cicli A.S.D.;controlloStart controlloA controlloB controlloEnd
```