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

Prova d'Esame 2A di Giovedì 28 Gennaio 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"*.

L'associazione di squadre di calcio professionistiche organizza ogni anno il torneo principale. Ogni domenica vengono giocate le partite, e i risultati vengono scritti su un file di testo, una partita per ogni riga. Vengono memorizzate le seguenti informazioni: la **data** della partita (nel formato **gg/mm/aaaa**, tre interi); a seguire, separata da uno spazio, il nome della squadra che gioca "**in casa**" (una stringa di al più 1023 caratteri utili, con spazi); separata da un carattere '-', il nome della squadra che gioca "**fuori casa**" (una stringa di al più 1023 caratteri utili, con spazi); infine, dopo un carattere ';' di separazione, due interi rappresentati i **gol** fatti dalla **prima** e dalla **seconda** squadra (i due interi separati semplicemente da uno spazio). Si noti che nel file con i risultati non è noto a priori se l'ultima riga sia terminata o meno da un carattere 'newline'.
In un secondo file di testo viene memorizzata la classifica: in ogni riga vi è il **nome** della squadra, e separati da un carattere ';' i **punti** attuali (un intero). Le squadre sono memorizzate nel file in un ordine casuale. Anche in questo file non è noto a priori se l'ultima riga sia terminata o meno da un carattere 'newline'.
Si vedano, a titolo di esempio, i file di testo "partite.txt" e "classifica.txt" forniti nello StartKit.

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

Si definisca un'opportuna struttura dati **`Data`**, al fine di rappresentare una data definita come giorno, mese e anno (tre interi). Si definisca un'opportuna struttura dati **`Partita`**, al fine di rappresentare le informazioni relative ad una partita (data, nome squadra in casa, nome squadra fuori casa, goal prima squadra, goal seconda squadra). Si definisca infine la struttura dati **`Punteggio`**, al fine di rappresentare i dati relativi al punteggio di una squadra (nome squadra, punti attuali).
Si definisca la funzione:

```
Partita * leggiPartite(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente le informazioni relative alle partite, legga da tale file tali informazioni, e le restituisca tramite un array allocato dinamicamente (di dimensione minima possibile) di strutture dati di tipo **`Partita`**. Tramite il parametro **`dim`** passato per riferimento la funzione deve restituire la dimensione dell'array allocato dinamicamente. Si assuma per semplicità che il file di input sia sempre ben formato secondo le specifiche date sopra, e l'unica possibile condizione non nota sia la presenza o assenza del carattere 'newline' nell'ultima riga del file. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve restituire un puntatore a NULL e come valore della dimensione il valore zero.
Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file di testo contenente la classifica attuale, restituisca una lista di strutture dati di tipo **`Punteggio`**, contenente le informazioni presenti nel file il cui nome è passato come parametro (nome squadra, punti). Si assuma per semplicità che il file di input sia sempre ben formato secondo le specifiche date sopra, e l'unica possibile condizione non nota sia la presenza o assenza del carattere 'newline' nell'ultima riga del file. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve restituire una lista vuota.
Si definisca la procedura **ricorsiva**:

```
void stampaPartite(Partita * v, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Partita`**, e la sua dimensione **`dim`**, stampi a video i risultati delle partite.

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 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

Esercizio 2 – Ordinamento (moduli element.h/.c e calcio.h/.c)

Il candidato definisca una procedura:

```
void ordinaPartite(Partita * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Partita`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine crescente in base ai gol fatti dalla squadra che gioca in casa; a parità di goal, in ordine crescente in base ai gol fatti dalla squadra che gioca fuori casa. A tal fine, il candidato utilizzi l'algoritmo "merge sort" visto a lezione.

Il candidato definisca una procedura

```
list ordinaClassifica(list classifica);
```

che, ricevuti in ingresso una lista di strutture dati di tipo **`Punteggio`**, crei una nuova lista contenente i dati in ingresso, e ordinata in modo decrescente rispetto al punteggio (le squadre con più punti devono comparire prima).

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

Esercizio 3 – Eliminazione duplicati e aggiornamento della classifica (modulo calcio.h/calcio.c)

Si vuole realizzare una funzionalità che permetta, dati i risultati delle varie partite, di aggiornare la classifica secondo la seguente regola: alla squadra vincitrice vengono assegnati 3 punti; in caso di pareggio, viene assegnato un punto a ciascuna squadra.
Si definisca dapprima una funzione:

```
int calcolaPremio(Partita v, char * nomeSquadra);
```

che ricevuto in ingresso una struttura dati di tipo Partita, e il nome di una squadra, restituisca il valore 3 (tre punti) se la squadra indicata ha vinto la partita, valore 1 (un punto) se la squadra ha pareggiato la partita, 0 (zero punti) altrimenti (cioè se ha perso).

Il candidato definisca la funzione:

```
list aggiornaClassifica(Partita * v, int dim, list classificaVecchia);
```

che, ricevuti come parametri un vettore di strutture dati di tipo **`Partita`** e la sua dimensione, e una lista di strutture dati di tipo **`Punteggio`**, restituisca una nuova lista con la classifica (cioè i punti) aggiornata come spiegato sopra.

Esercizio 4 – Stampa della classifica aggiornata, 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 la classifica aggiornata del campionato di calcio: l'aggiornamento dovrà avvenire usando i risultati delle partite forniti nello StartKit di esempio.

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 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

typedef struct {
	int gg;
	int mm;
	int aaaa;
} Data;


typedef struct {
	Data data;
	char inCasa[1024];
	char fuoriCasa[1024];
	int golPrima;
	int golSeconda;
} Partita;

typedef struct {
	char nome[1024];
	int punti;
} Punteggio;

typedef Punteggio element;


int comparePartita(Partita p1, Partita p2);
int comparePunteggio(Punteggio p1, Punteggio p2);

#endif
```

"element.c":

```
#include "element.h"


int comparePartita(Partita p1, Partita p2) {
	int result;
	result = p1.golPrima - p2.golPrima;
	if (result == 0)
		result = p1.golSeconda - p2.golSeconda;
	return result;
}

int comparePunteggio(Punteggio p1, Punteggio p2) {
	return p2.punti - p1.punti;
}
```

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

Prova d'Esame 2A di Giovedì 28 Gennaio 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 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 (comparePunteggio(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 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

"calcio.h":

```
#ifndef _CALCIO_H
#define _CALCIO_H

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

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

Partita * leggiPartite(char * fileName, int * dim);
list leggiClassifica(char* fileName);
void stampaPartite(Partita * v, int dim);
void stampaClassifica(list classifica);

void ordinaPartite(Partita * v, int dim);
list ordinaClassifica(list classifica);

int calcolaPremio(Partita v, char * nomeSquadra);
list aggiornaClassifica(Partita * v, int dim, list classificaVecchia);

#endif
```

"calcio.c":

```
#include "calcio.h"

int readField(char buffer[], int dimBuffer, char sep, FILE *f) {
	int i = 0;
	char ch = fgetc(f);
	while (ch != sep && ch != '\n' && ch != EOF && i<dimBuffer-1) {
		buffer[i] = ch;
		i++;
		ch = fgetc(f);
	}
	buffer[i] = '\0';
	return ch;
}

Partita * leggiPartite(char * fileName, int * dim) {
	FILE * fp;
	Partita * result;
	Partita temp;
	int i;
	int okLettura;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		result = NULL;
	}
	else {
		okLettura = 1;
```

```
            while (fscanf(fp, "%d/%d/%d", &(temp.data.gg), &(temp.data.mm),
&(temp.data.aaaa)) == 3 && okLettura ) {
                    fgetc(fp);
                    okLettura = (readField(temp.inCasa, 1024, '-', fp) == '-');
                    if (okLettura)
                           okLettura = (readField(temp.fuoriCasa, 1024, ';', fp) == ';');
                    if (okLettura)
                           okLettura = (fscanf(fp, "%d%d", &(temp.golPrima),
&(temp.golSeconda)) == 2);
                    if (okLettura)
                           *dim = *dim + 1;
             }
             if (*dim>0) {
                    result = (Partita *) malloc(sizeof(Partita) * *dim);
                    rewind(fp);
                    okLettura = 1;
                    i = 0;
                    while (fscanf(fp, "%d/%d/%d", &(temp.data.gg), &(temp.data.mm),
&(temp.data.aaaa)) == 3 && okLettura ) {
                           fgetc(fp);
                           okLettura = (readField(temp.inCasa, 1024, '-', fp) == '-');
                           if (okLettura)
                                  okLettura = (readField(temp.fuoriCasa, 1024, ';', fp) ==
';');
                           if (okLettura)
                                  okLettura = (fscanf(fp, "%d%d", &(temp.golPrima),
&(temp.golSeconda)) == 2);
                           if (okLettura) {
                                  result[i] = temp;
                                  i++;
                           }
                    }
             }
             else
                    result = NULL;
             fclose(fp);
       }
       return result;
}

list leggiClassifica(char* fileName) {
       FILE * fp;
       list result;
       Punteggio temp;
       int okLettura;

       result = emptylist();
       fp = fopen(fileName, "rt");

       if (fp != NULL) {
              okLettura = 1;
              while (okLettura) {
                    okLettura = (readField(temp.nome, 1024, ';', fp) == ';');
                    if (okLettura)
                           okLettura = (fscanf(fp, "%d", &(temp.punti) ) == 1);
                    if (okLettura) {
                           result = cons(temp, result);
```

```
                        fgetc(fp);
                }
            }
            fclose(fp);
        }
        return result;
}

void stampaPartite(Partita * v, int dim) {
        Partita temp;
        if (dim <= 0)
                return;
        else {
                stampaPartite(v, dim-1);
                temp = v[dim-1];
                printf("%d/%d/%d %s-%s %d-%d\n",
                        temp.data.gg, temp.data.mm, temp.data.aaaa,
                        temp.inCasa, temp.fuoriCasa, temp.golPrima, temp.golSeconda);
                return;
        }
}

void stampaClassifica(list classifica) {
        Punteggio temp;
        if (empty(classifica))
                return;
        else {
                temp = head(classifica);
                printf("%s %d\n", temp.nome, temp.punti);
                stampaClassifica(tail(classifica));
                return;
        }
}

void merge(Partita v[], int i1, int i2, int fine, Partita vout[]){
        int i=i1, j=i2, k=i1;
        while ( i <= i2-1 && j <= fine ) {
                if (comparePartita(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(Partita v[], int first, int last, Partita 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);
        }
}
```

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

Prova d'Esame 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

```
void ordinaPartite(Partita * v, int dim) {
	Partita * temp;
	temp = (Partita *) malloc(sizeof(Partita) * dim);
	mergeSort(v, 0, dim-1, temp);
	free(temp);
	return;
}

list ordinaClassifica(list classifica) {
	Punteggio temp;
	list result = emptylist();

	while(!empty(classifica)) {
		temp = head(classifica);
		result = insord_p(temp, result);
		classifica = tail(classifica);
	}
	return result;
}

int calcolaPremio(Partita v, char * nomeSquadra) {
	int result;
	if (strcmp(nomeSquadra, v.inCasa) == 0) {
		if (v.golPrima>v.golSeconda)
			result = 3;
		else if (v.golPrima == v.golSeconda)
			result = 1;
			else
				result = 0;
	}
	else
		if (strcmp(nomeSquadra, v.fuoriCasa) == 0) {
			if (v.golSeconda > v.golPrima)
				result = 3;
			else
				if (v.golPrima == v.golSeconda)
					result = 1;
				else
					result = 0;
		}
		else {
			printf("Partita sbagliata, la squadra richiesta non compare!!!\n");
			result = -100;
		}
	return result;
}

list aggiornaClassifica(Partita * v, int dim, list classificaVecchia) {
	list nuovaClassifica;
	Punteggio tempPunt;
	int trovato;
	int i;
	int pos;
```

```
	nuovaClassifica = emptylist();
	while (!empty(classificaVecchia)) {
		tempPunt = head(classificaVecchia);
		trovato = 0;
		for (i=0; i<dim && !trovato; i++) {
			if (strcmp(tempPunt.nome, v[i].inCasa)==0 ||
strcmp(tempPunt.nome,v[i].fuoriCasa)==0) {
				trovato = 1;
				pos = i;
			}
		}
		if (trovato) {
			tempPunt.punti = tempPunt.punti + calcolaPremio(v[pos],
tempPunt.nome);
		}
		else
			printf("Partita relativa a %s non trovata!!!\n", tempPunt.nome);
		nuovaClassifica = cons(tempPunt, nuovaClassifica);
		classificaVecchia = tail(classificaVecchia);
	}
	return nuovaClassifica;
}
```

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

Prova d'Esame 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "list.h"
#include "calcio.h"

int main() {
	{ // Es. 1
		Partita * giornata;
		int dim;
		list classifica;
		giornata = leggiPartite("partite.txt", &dim);
		stampaPartite(giornata, dim);
		classifica = leggiClassifica("classifica.txt");
		stampaClassifica(classifica);
		free(giornata);
		freelist(classifica);
	}
	{ // Es. 2
		Partita * giornata;
		int dim;
		list classifica;
		list classificaOrd;

		giornata = leggiPartite("partite.txt", &dim);
		classifica = leggiClassifica("classifica.txt");
		ordinaPartite(giornata, dim);
		classificaOrd = ordinaClassifica(classifica);

		stampaPartite(giornata, dim);
		stampaClassifica(classificaOrd);

		free(giornata);
		freelist(classifica);
		freelist(classificaOrd);
	}
	{ // Es. 3 && 4
		Partita * giornata;
		int dim;
		list classifica;
		list nuovaClassifica;
		list classificaOrd;

		giornata = leggiPartite("partite.txt", &dim);
		classifica = leggiClassifica("classifica.txt");
		nuovaClassifica = aggiornaClassifica(giornata, dim, classifica);
		classificaOrd = ordinaClassifica(nuovaClassifica);

		stampaClassifica(classificaOrd);

		free(giornata);
		freelist(classifica);
		freelist(nuovaClassifica);
		freelist(classificaOrd);
	}
	return 0;
}
```

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

Prova d'Esame 2A di Giovedì 28 Gennaio 2016 – tempo a disposizione 2h

"partita.txt":

```
24/01/2016 Frosinone-Atalanta;0 0
24/01/2016 Empoli-Milan;2 2
24/01/2016 Fiorentina-Torino;2 0
24/01/2016 Inter-Carpi;1 1
24/01/2016 Lazio-Chievo;4 1
24/01/2016 Palermo-Udinese;4 1
24/01/2016 Sampdoria-Napoli;2 4
24/01/2016 Sassuolo-Bologna;0 2
24/01/2016 Verona-Genoa;1 1
24/01/2016 Juventus-Roma;1 0
```

"classifica.txt":

```
Napoli;47
Juventus;45
Fiorentina;41
Inter;41
Roma;35
Milan;33
Empoli;32
Sassuolo;32
Lazio;31
Chievo;27
Atalanta;26
Bologna;26
Torino;26
Palermo;24
Udinese;24
Genoa;23
Sampdoria;23
Carpi;18
Frosinone;16
Verona;10
```