Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit1A.zip`** contenente i file necessari (*progetto Visual Studio* ed eventuali altri file di esempio).
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"*.

La Comunità Europea, tra i vari compiti istituzionali, ha anche quello di finanziare periodicamente dei progetti di ricerca. A tal scopo, durante certi periodi dell'anno, è possibile inviare delle richieste di finanziamento. Tali richieste sono valutate da tre esperti indipendenti, ognuno dei quali fornisce un voto. Al termine della valutazione a ogni progetto viene assegnato un voto totale dato dalla somma dei voti di ogni valutatore, e poi vengono finanziati i progetti migliori.

In un file di testo, denominato "**`progetti.txt`**", viene salvato l'elenco dei progetti. Nella prima riga del file è contenuto il numero di progetti memorizzati in tale file. Poi nelle righe successive, su ogni riga, viene salvato **l'identificativo** numerico di un progetto (un intero), il **finanziamento** richiesto (un float, separato da uno spazio dal campo precedente), e a seguire il **titolo** del progetto (una stringa di al più 1023 caratteri, contenente spazi). Ogni riga (compresa l'ultima riga) è terminata sempre da un carattere "newline".

In un secondo file di testo denominato "**`valutazioni.txt`**" sono memorizzati i voti che ogni valutatore ha dato ad ogni progetto valutato. In particolare, su ogni riga, è memorizzato il **nome** del valutatore (una stringa di al più 23 caratteri, senza spazi), il codice **identificativo** del progetto valutato, e infine il **voto** assegnato (un intero, il cui valore è sempre compreso tra 1 e 5); ogni campo è separato dagli tramite uno spazio.

Esercizio 1 – Strutture dati Progetto e funzioni di lettura/scrittura (mod. element.h e progetti.h/progetti.c)

Si definisca un'opportuna struttura dati **`Progetto`**, al fine di rappresentare i dati relativi ad ogni progetto: in particolare si dovrà tenere traccia del codice identificativo, del titolo del progetto, dell'importo richiesto, e dei tre voti ricevuti. Si definisca poi una opportuna struttura dati **`Valutazione`**, al fine di rappresentare una singola valutazione di un singolo progetto: quindi si dovrà tenere traccia del nome del valutatore, dell'identificativo del progetto, e del voto.
Si definisca la funzione:

```
Progetto * leggiProgetti(char* fileName, int *dim);
```

che, ricevuto in ingresso il nome di un file di testo contenente l'elenco dei progetti, restituisca un array di strutture dati di tipo **`Progetto`** allocato dinamicamente (della dimensione minima necessaria), contenente tutte le informazioni presenti nel file il cui nome è passato come parametro. Tramite il parametro **`dim`** la funzione deve restituire la dimensione del vettore. Siccome tale funzione non ha accesso ai voti assegnati al progetto, i valori dei voti assegnati dovranno essere inizializzati a "-1".
Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file di testo contenente le valutazioni, restituisca una lista di strutture dati di tipo **`Valutazione`**.
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, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

Esercizio 2 – "merge" delle informazioni e ordinamento (moduli element.h/.c e progetti.h/.c)

Il candidato definisca una procedura:

```
void merge(Progetto * v, int dim, list valutazioni);
```

tale che, ricevuto un vettore **v** di strutture dati di tipo **Progetto**, la dimensione di tale vettore, e la lista delle valutazioni, assegni ad ogni progetto in v i voti relativi. Si noti che può accadere che qualche valutatore sia in ritardo, e che quindi per qualche progetto siano disponibili solo due valutazioni (o anche meno) e non tre come previsto. In tal caso, il voto deve essere posto pari a "-1" (se non lo è già).

Il candidato definisca poi una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **Progetto** e la dimensione di tale vettore, ordini il vettore in maniera decrescente (dal più grande al più piccolo) in base al punteggio totale assegnato ad ogni progetto. Il punteggio totale è dato dalla somma dei tre singoli voti.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Esercizio 3 –Scelta dei progetti da finanziare (modulo progetti.h/progetti.c)

Una volta noti i progetti e i loro punteggi totali, è necessario decidere quali finanziare. Ovviamente verranno finanziati i progetti con punteggio totale migliore, fino ad esaurimento dei fondi disponibili. Inoltre, per garantire correttezza, non possono essere finanziati i progetti per cui uno o più voti non siano noti (cioè valgano ancora "-1"). A tal scopo, il candidato definisca una funzione:

```
Progetto * finanziati(Progetto * v, int dimV, int * dimR, float soldi);
```

che, ricevuti come parametri un vettore di strutture dati di tipo **Progetto** e la sua dimensione **dimV**, restituisca un nuovo vettore di strutture dati di tipo **Progetto** e la sua dimensione **dimR** (non necessariamente della dimensione minima), dove siano stati inseriti i progetti finanziabili. Si abbia cura che i progetti selezionati per il finanziamento non abbiano nessun voto pari a "-1"; inoltre l'ammontare di tutti i finanziamenti concessi non deve andare oltre il budget disponibile, che è specificato tramite il parametro **soldi**.

Ad esempio se la funzione viene invocata usando come dati di esempio quelli contenuti nello startkit, e con budgt totale disponibile pari a 5 milioni di euro, risulteranno essere finanziati i soli progetti con identificativo "27" e "56".

Esercizio 4 – Stampa dei progetti finanziabili, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **main(…)** un programma che, usando le informazioni fornite tramite i file di esempio forniti nello StartKit, e le funzioni definite agli esercizi precedenti, chieda all'utente di definire l'ammontare di finanziamento disponibile, e stampi a video l'elenco dei progetti finanziabili.

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, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define	_ELEMENT_H

#define TITLE_SIZE 1024
#define NAME_SIZE 24


typedef struct {
	int id;
	char title[TITLE_SIZE];
	float budget;
	int score[3];
} Progetto;

typedef struct {
	char nome[NAME_SIZE];
	int id;
	int score;
} Valutazione;


typedef Valutazione element;

int compareProgetto(Progetto p1, Progetto p2);



#endif /* _ELEMENT_H */
```

"element.c":

```
#include "element.h"


int compareProgetto(Progetto p1, Progetto p2) {
	int j;
	int score1=0, score2=0;

	for (j=0; j<3; j++)
		score1 = score1 + p1.score[j];
	for (j=0; j<3; j++)
		score2 = score2 + p2.score[j];
	return score2-score1;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – 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;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

"progetti.h":

```
#ifndef _PROGETTI_H
#define _PROGETTI_H

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

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


Progetto * leggiProgetti(char* fileName, int *dim);
list leggiValutazioni(char* fileName);

void merge(Progetto * v, int dim, list valutazioni);
void ordina(Progetto * v, int dim);

Progetto * finanziati(Progetto * v, int dimV, int * dimR, float soldi);
#endif
```

"progetti.c":

```
#include "progetti.h"


Progetto * leggiProgetti(char* fileName, int *dim) {
	FILE * fp;
	Progetto * result = NULL;
	Progetto temp;
	int i, j;
	char ch;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore durante l'apertura del file: %s\n\n", fileName);
		system("pause");
		exit(-1);
	}
	else {
		fscanf(fp, "%d", dim);
		result = (Progetto*) malloc(sizeof(Progetto) * *dim);

		for (i=0; i<*dim; i++) {
			fscanf(fp, "%d%f", &(temp.id), &(temp.budget));
			j = 0;
			while ( (ch=fgetc(fp)) != '\n') {
				temp.title[j] = ch;
				j++;
			}
			temp.title[j] = '\0';
			for (j=0; j<3; j++)
				temp.score[j] = -1;
```

```
                result[i] = temp;
            }
            fclose(fp);
            return result;
        }
}

list leggiValutazioni(char* fileName) {
        FILE * fp;
        Valutazione temp;
        list result;

        result = emptylist();
        fp = fopen(fileName, "rt");
        if (fp == NULL) {
            printf("Errore durante l'apertura del file: %s\n\n", fileName);
            system("pause");
            exit(-1);
        }
        else {
            while( fscanf(fp, "%s%d%d", temp.nome, &(temp.id), &(temp.score)) == 3 )
                result = cons(temp, result);
            fclose(fp);
            return result;
        }
}

void merge(Progetto * v, int dim, list valutazioni) {
        int i, j;
        list temp;

        for (i=0; i<dim; i++) {
            temp = valutazioni;
            for (j=0; j<3 && !empty(temp); j++) {
                while (!empty(temp) && head(temp).id != v[i].id)
                    temp = tail(temp);
                if (!empty(temp)) {
                    v[i].score[j] = head(temp).score;
                    temp = tail(temp);
                }
            }
        }
        return;
}

void scambia(Progetto *a, Progetto *b) {
        Progetto tmp = *a;
        *a = *b;
        *b = tmp;
}
// bubble sort
void ordina(Progetto * v, int n) {
        int i, ordinato = 0;
        while (n>1 && !ordinato) {
            ordinato = 1;
            for (i=0; i<n-1; i++)
                if (compareProgetto(v[i],v[i+1])>0) {
```

```
                        scambia(&v[i],&v[i+1]);
                        ordinato = 0;
                }
            n--;
        }
}

int finanziabile(Progetto p) {
        int i;
        int result = 1;

        for (i=0; i<3 && result; i++)
            if (p.score[i]<=0)
                result = 0;
        return result;
}

// Questa versione finanzia fino ad esaurimento budget, escludendo eventualmente progetti
troppi costosi...
Progetto * finanziati(Progetto * v, int dimV, int * dimR, float soldi) {
        Progetto * result;
        int i;
        float total = 0.0f;
        *dimR = 0;
        result = (Progetto*) malloc(sizeof(Progetto) * dimV);
        for (i=0; i<dimV && total<soldi; i++) {
            if (finanziabile(v[i]) && total+v[i].budget <= soldi) {
                result[*dimR] = v[i];
                (*dimR)++;
                total = total + v[i].budget;
            }
        }
        return result;
}

// Questa versione invece finanzia sempre rispettando l'ordine, ed eventualmente non
assegna dei fondi...
Progetto * finanziati2(Progetto * v, int dimV, int * dimR, float soldi) {
        Progetto * result;
        int i;
        float total = 0.0f;
        *dimR = 0;
        result = (Progetto*) malloc(sizeof(Progetto) * dimV);
        for (i=0; i<dimV && total<soldi; i++) {
            if ( finanziabile(v[i]) )
                if (total+v[i].budget <= soldi) {
                    result[*dimR] = v[i];
                    (*dimR)++;
                    total = total + v[i].budget;
                }
                else
                    total = soldi; //non posso piu' finanziare altri progetti...
            }
        }
        return result;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{ // test es. 1
		Progetto * elenco;
		list valutazioni;
		int dim_elenco;
		int i, j;

		elenco = leggiProgetti("progetti.txt", &dim_elenco);
		for (i=0; i<dim_elenco; i++) {
			printf("%d %f %s\n", elenco[i].id, elenco[i].budget, elenco[i].title);
			printf("Scores: ");
			for (j=0; j<3; j++)
				printf("%d ", elenco[i].score[j]);
			printf("\n");
		}
		valutazioni = leggiValutazioni("valutazioni.txt");
		while (!empty(valutazioni)) {
			Valutazione temp = head(valutazioni);
			printf("%s %d %d\n", temp.nome, temp.id, temp.score);
			valutazioni = tail(valutazioni);
		}
		free(elenco);
		freelist(valutazioni);
	}
	{ // test es. 2
		Progetto * elenco;
		list valutazioni;
		int dim_elenco;
		int i, j;

		elenco = leggiProgetti("progetti.txt", &dim_elenco);
		valutazioni = leggiValutazioni("valutazioni.txt");
		merge(elenco, dim_elenco, valutazioni);
		ordina(elenco, dim_elenco);
		for (i=0; i<dim_elenco; i++) {
			printf("%d %f %s\n", elenco[i].id, elenco[i].budget, elenco[i].title);
			printf("Scores: ");
			for (j=0; j<3; j++)
				printf("%d ", elenco[i].score[j]);
			printf("\n");
		}

		free(elenco);
		freelist(valutazioni);
	}
	{ // test es. 3 e 4
		Progetto * elenco;
		Progetto * finanziabili;
```

```
		list valutazioni;
		int dim_elenco;
		int dim_fin;
		int i, j;
		float soldi;

		elenco = leggiProgetti("progetti.txt", &dim_elenco);
		valutazioni = leggiValutazioni("valutazioni.txt");
		merge(elenco, dim_elenco, valutazioni);
		ordina(elenco, dim_elenco);
		printf("Inserire l'ammontare di finanziamento disponibile: ");
		scanf("%f", &soldi);
		finanziabili = finanziati(elenco, dim_elenco, &dim_fin, soldi);
		printf("\n\n=== FINANZIABILI ===\n");
		for (i=0; i<dim_fin; i++) {
			printf("%d %f %s\n", finanziabili[i].id, finanziabili[i].budget,
finanziabili[i].title);
			printf("Scores: ");
			for (j=0; j<3; j++)
				printf("%d ", finanziabili[i].score[j]);
			printf("\n");
		}

		free(elenco);
		free(finanziabili);
		freelist(valutazioni);
	}



	system("pause");
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 1A di Giovedì 10 Gennaio 2013 – tempo a disposizione 2h

"progetti.txt":

```
4
27 1300450.00 Un progetto per rendere migliore il mondo
34 2500670.34 Una nuova infrastruttura per AlmaWifi
56 3400780.00 Modelli e strumenti per la previsione della fine del mondo
99 4600340.00 I Maya si sono sbagliati? Una rilettura critica del loro calendario
```

"valutazioni.txt":

```
Chesani 27 4
Chesani 34 2
Giannelli 27 5
Bragaglia 27 5
Bragaglia 34 3
Chesani 56 4
Giannelli 34 3
Chesani 99 4
Giannelli 56 4
Giannelli 99 4
Bragaglia 56 5
```