# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – 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"*.

Un'importante università italiana ha deciso di limitare il numero massimo di studenti che si vogliano iscrivere ad alcuni corsi universitari. Per decidere chi si possa iscrivere, vengono effettuati dei test che verificano le competenze su diverse aree. Gli studenti possono ritentare i test più volte, e ai fini della selezione si considera il test con il punteggio totale migliore. Tutti i risultati dei test sono memorizzati in un unico file di testo.
In tale file in particolare per ogni studente e per ogni prova vengono memorizzate le seguenti informazioni: la **data** di sostenimento del test (formato gg/mm/aaaa, tre interi); separato da uno spazio, la **matricola** dello studente (una stringa di al più 15 caratteri, senza spazi); infine, ancora separato da uno spazio, un elenco di quattro coppie <materia, punti> (materia una stringa senza spazi di al più 15 caratteri utili, punteggio un intero), ogni coppia separata da spazi, e internamente ancora uno spazio di separazione.
Il file di testo ha una peculiarità: i dati relativi ad ogni studente e ad ogni prova sono tutti registrati su un'unica riga, e sono separati tra loro da un carattere '@' (l'ultimo elemento a volte è seguito dal carattere '@', a volte no). Si veda, per esempio, il file fornito nello StartKit.
In un secondo file di testo sono memorizzate le preferenze di uno studente: su ogni riga di tale file è memorizzata la matricola e, separato da uno spazio il corso di studi che vuole frequentare lo studente (una stringa di al più 15 caratteri utili).

### *Esercizio 1 – Strutture dati Data, Punteggio, e Test, e funzioni di lett./scritt. (mod. element.h e test.h/.c)*

Si definiscano opportune strutture dati **`Data`** (per memorizzare una data), **`Punteggio`** (per memorizzare il nome di una materia ed il punteggio preso) e **`Test`** (per memorizzare l'esito di un test sostenuto da uno studente).

Si definisca la funzione:

```
Test leggiUnTest(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo **`FILE`**, contenente i dati relativi ai test, legga i dati contenuti relativi ad un solo test, e li restituisca tramite una struttura dati di tipo **`Test`**. Si assuma per semplicità che tutti dati siano sempre presenti e memorizzati correttamente nel file. Qualora si sia giunti al termine del file, o non sia possibile leggere, la funzione deve restituire una struttura dati con valore del campo matricola pari alla stringa "**`NULL`**".
Si definisca la funzione:

```
Test * leggiTutti(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga da tale file tutte le informazioni relative ai test e le restituisca tramite un array di strutture dati di tipo **`Test`** allocato dinamicamente (della dimensione minima necessaria). Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore. Qualora la funzione incontri dei problemi nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a **`NULL`**, e zero come valore della dimensione.
Si definisca la procedura:

```
void stampaTest(Test * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Test`**, e la sua dimensione **`dim`**, stampi a video le informazioni riguardo i test memorizzati.
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, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

### *Esercizio 2 –Ordinamento e lista delle preferenze (moduli element.h/.c e test.h/.c)*

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Test`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine decrescente in base al punteggio totale ottenuto nel test (i test col punteggio totale maggiore vengono *prima* nel vettore). Il punteggio totale è determinato come somma dei singoli punteggi. A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "bubble sort" visto a lezione.

Si definisca una opportuna struttura dati **`Preferenza`**, per memorizzare matricola e corso di studi indicato da uno studente.

Si definisca poi una funzione:

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

che, ricevuto in ingresso il nome di un file contenente le preferenze per i corsi di studio, restituisca una lista di strutture dati di tipo **`Preferenza`**, contenente le coppie matricola – corso. Qualora vi siano problemi di lettura o apertura del file, la funzione deve restituire una lista vuota. Si noti che qualche volta uno studente indica più preferenze, che vengono registrate tutte nel file. Nella lista dovrà comparire solo la prima preferenza registrata nel file, mentre le altre saranno ignorate.

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.

### *Esercizio 3 – Elenco degli ammessi (modulo test.h/test.c)*

L'università necessita di sapere quali studenti saranno ammessi ai corsi di laurea indicati, e quali no. A tal scopo, si definisca una procedura:

```
void ammessi(Test * v, int dim, list elenco, int postiDisponibili, char* corso);
```

che ricevuti come parametri un vettore di strutture dati di tipo **`Test`** e la sua dimensione, e una lista di strutture dati di tipo **`Preferenza`**, e il numero massimo di posti disponibili per un corso (pure specificato), stampi a video le matricole degli studenti che sono ammessi al corso specificato. Il criterio di ammissione è il seguente: vengono ammessi al corso, fino ad esaurimento posti, gli studenti che hanno espresso la preferenza per quel corso, e il cui punteggio totale nel test è migliore. Il punteggio totale è dato dalla somma dei singoli punteggi.

**Suggerimento**: si definisca una funzione di utilità che, data la matricola di uno studente e il vettore di tutti i test, restituisca il test migliore fatto dalla matricola specificata. Si usi poi un vettore temporaneo di appoggio (allocato dinamicamente) per memorizzare tutti i test migliori fatti dagli studenti che hanno fatto richiesta per un certo corso.

### *Esercizio 4 Stampa degli ammessi, e de-allocazione memoria (main.c)*

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file i dati relativi ai test e alle preferenze, e stampi a video l'elenco degli ammessi ad un certo corso di laurea. Al fine di controllare la corretta esecuzione del programma, si specifichi come corso "Ingegneria", e come numero di posti disponibili il valore cinque.

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, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

#include <string.h>
#define DIM 16

typedef struct {
	int gg;
	int mm;
	int aa;
} Data;
typedef struct {
	char materia[DIM];
	int punti;
} Punteggio;
typedef struct {
	Data data;
	char matricola[DIM];
	Punteggio punti[4];
} Test;
typedef struct {
	char matricola[DIM];
	char corso[DIM];
} Preferenza;
typedef Preferenza element;

int compare(Test t1, Test t2);
int equals(Preferenza p1, Preferenza p2);
int punt(Test t);

#endif
```

"element.c":

```
#include "element.h"

int compare(Test t1, Test t2) {
	int p1=0;
	int p2=0;
	p1 = punt(t1);
	p2 = punt(t2);
	return p2-p1;
}

int equals(Preferenza p1, Preferenza p2) { return !strcmp(p1.matricola, p2.matricola); }

int punt(Test t) {
	int i;
	int result = 0;
	for (i=0; i<4; i++) {
		result = result + t.punti[i].punti;
	}
	return result;
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – 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 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);
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

```
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\n",
                     temp.matricola, temp.corso);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}

int member(element el, list l) {
       int result = 0;
       while (!empty(l) && !result) {
              result = equals(el, head(l));
              if (!result)
                     l = tail(l);
       }
       return result;
}

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

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

"test.h":

```
#ifndef _TEST_H
#define _TEST_H

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

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

// Es. 1
Test leggiUnTest(FILE * fp);
Test * leggiTutti(char * fileName, int * dim);
void stampaTest(Test * v, int dim);

// Es. 2
void ordina(Test * v, int dim);
list pref(char * fileName);

// Es. 3
void ammessi(Test * v, int dim, list elenco, int postiDisponibili, char* corso);
#endif
```

"test.c":

```
#include "test.h"

// Es. 1
Test leggiUnTest(FILE * fp) {
	Test result;
	int success;
	int i;

	strcpy(result.matricola, "NULL");
	if (fp!= NULL) {
		success = fscanf(fp, "%d/%d/%d %s",
			&(result.data.gg), &(result.data.mm), &(result.data.aa),
result.matricola);
		if (success == 4) {
			success = 2;
			for (i=0; success==2 && i<4; i++) {
				success = fscanf(fp, "%s%d", result.punti[i].materia,
&(result.punti[i].punti) );
				if (success!=2)
					strcpy(result.matricola, "NULL");
			}
		}
		else {
			strcpy(result.matricola, "NULL");
		}
	}
	return result;
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

```
Test * leggiTutti(char * fileName, int * dim) {
	FILE * fp;
	Test * result;
	Test temp;
	int cont = 0;
	int i;

	*dim = 0;
	result = NULL;
	fp = fopen(fileName, "r");
	if (fp != NULL) {
		do {
			temp = leggiUnTest(fp);
			if (strcmp(temp.matricola, "NULL")!= 0)
				cont++;
			fgetc(fp);
		} while (strcmp(temp.matricola, "NULL")!= 0);

		result = (Test*) malloc(sizeof(Test) * cont);
		rewind(fp);
		for (i=0; i<cont; i++) {
			temp = leggiUnTest(fp);
			fgetc(fp);
			result[i] = temp;
		}
		*dim = i;
		fclose(fp);
	}
	else {
		printf("Errore in apertura del file %s\n", fileName);
	}

	return result;
}

void stampaTest(Test * v, int dim) {
	int i;
	int j;

	for (i=0; i<dim; i++) {
		printf("%d/%d/%d %s\n", v[i].data.gg, v[i].data.mm, v[i].data.aa,
v[i].matricola);
		for (j=0; j<4; j++) {
			printf("\t%s %d\n", v[i].punti[j].materia, v[i].punti[j].punti);
		}
		printf("\n");
	}
	return;
}

// Es. 2
void scambia(Test *a, Test *b) {
	Test tmp = *a;
	*a = *b;
	*b = tmp;
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

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

void ordina(Test * v, int dim) { bubbleSort(v, dim); }

list pref(char * fileName) {
	FILE * fp;
	list result;
	Preferenza temp;

	result = emptylist();
	fp = fopen(fileName, "r");
	if (fp!=NULL) {
		while(fscanf(fp, "%s%s", temp.matricola, temp.corso)==2) {
			if (!member(temp, result))
				result = cons(temp, result);
		}
		fclose(fp);
	}
	return result;
}

Test findBest(Test * v, int dim, char * matricola) {
	int i;
	Test result;
	int best = 0;

	for (i=0; i<dim; i++) {
		if (strcmp(matricola, v[i].matricola)==0) {
			if (punt(v[i])>best) {
				best = punt(v[i]);
				result = v[i];
			}
		}
	}
	return result;
}

// Es. 3
void ammessi(Test * v, int dim, list elenco, int postiDisponibili, char* corso) {
	Test * best;
	int dimBest = 0;
	int i;
	Preferenza p;

	best = (Test*) malloc(sizeof(Test)*dim);
```

```
        while(!empty(elenco)) {
                p = head(elenco);
                if (strcmp(p.corso, corso) == 0) {
                        best[dimBest] = findBest(v, dim, p.matricola);
                        dimBest++;
                }
                elenco = tail(elenco);
        }
        ordina(best, dimBest);
        printf("Elenco degli ammessi:\n");
        for (i=0; i<postiDisponibili && i<dimBest; i++) {
                printf("%s %d\n", best[i].matricola, punt(best[i]));
        }
        free(best);
        return;
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "test.h"

int main() {
	{	// Es. 1
		Test* v;
		int dim;

		v= leggiTutti("risultati.txt", &dim);
		stampaTest(v, dim);
		free(v);
	}

	{	// Es. 2
		Test* v;
		int dim;
		list l1;

		v= leggiTutti("risultati.txt", &dim);
		ordina(v, dim);
		printf("\n\n\n");
		stampaTest(v, dim);

		l1=pref("preferenze.txt");
		showlist(l1);

		freelist(l1);
		free(v);
	}

	{	// Es. 3
		Test* v;
		int dim;
		list l1;

		v= leggiTutti("risultati.txt", &dim);
		l1=pref("preferenze.txt");
		ammessi(v, dim, l1, 5, "Ingegneria");

		freelist(l1);
		free(v);
	}
	return 0;
}
```

# Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

## Prova d'Esame 3A di Giovedì 16 Febbraio 2017 – tempo a disposizione 2h

"risultati.txt":

```
16/02/2017 000123 logica 15 matematica 13 fisica 15 comprensione 9@16/02/2017 000234 logica
13 matematica 12 fisica 15 comprensione 11@16/02/2017 000345 logica 5 matematica 4 fisica 3
comprensione 8@16/02/2017 000456 logica 6 matematica 7 fisica 5 comprensione 12@18/02/2017
000123 logica 15 matematica 14 fisica 15 comprensione 14@18/02/2017 000234 logica 13
matematica 12 fisica 15 comprensione 11@26/03/2017 000123 logica 15 matematica 15 fisica 15
comprensione 15@26/03/2017 000345 logica 2 matematica 1 fisica 1 comprensione 10@26/03/2017
000567 logica 12 matematica 13 fisica 12 comprensione 9@26/03/2017 000678 logica 7 matematica
7 fisica 7 comprensione 7@26/03/2017 000789 logica 15 matematica 13 fisica 15 comprensione 9
```

"preferenze.txt"

```
000123 Ingegneria
000234 Ingegneria
000345 Ingegenria
000456 Ingegneria
000123 Medicina
000234 Lettere
000123 Lettere
000345 Agraria
000567 Ingegneria
000678 Ingegneria
000789 Ingegneria
```