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

## Prova d'Esame 2A di Martedì 29 Gennaio 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"*.

Una azienda fornisce i pasti per gli alunni di diverse scuole. L'azienda prepara in un file di testo un elenco di pietanze, con lista degli ingredienti contenuti. In particolare, su ogni riga del file, sono memorizzati nell'ordine i seguenti dati, tutti separati tra di loro da uno spazio: il **nome** di una pietanza (una stringa di al più 255 caratteri utili senza spazi); il **tipo** di pietanza, cioè un numero intero pari a uno o pari a due, a seconda che la pietanza sia da considerarsi un primo o un secondo; il **numero** di ingredienti di cui è composta la pietanza (un intero); gli **ingredienti**, separati tra loro da uno spazio, nel numero indicato (ogni ingrediente rappresentato come una stringa di al più 255 caratteri utili senza spazi). Non è noto a priori quante pietanze siano presenti nel file.
In un secondo file di testo sono memorizzate delle informazioni riguardo le intolleranze alimentari degli alunni, nella seguente forma: un **identificatore** unico dell'alunno (un intero); a seguire l'**elenco** di ingredienti (come sopra) a cui l'alunno specificato è intollerante. Ogni ingrediente è separato dal seguente da uno spazio; non è noto il numero di ingredienti a cui è intollerante ogni alunno, però l'ultimo ingrediente è sempre seguito da un carattere '\n', anche nell'ultima riga. Il file degli alunni però nella prima riga contiene solo un intero, indicante il numero di alunni memorizzati nel file.

Si vedano a titolo di esempio i file di testo forniti nello StartKit.

### *Esercizio 1 – Strutture dati Ingrediente, Pietanza, e funzioni di lett./scritt. (mod. element.h e mensa.h/.c)*

Si definisca un'opportuna struttura dati **`Ingrediente`**, atta a rappresentare un singolo ingrediente (per semplicità, il solo nome dell'ingrediente). Si definisca poi una opportuna struttura dati **`Pietanza`**, al fine di rappresentare il nome della pietanza, il tipo di pietanza, il numero e l'elenco degli ingredienti (tramite un vettore di strutture dati di tipo **`Ingrediente`**).

Si definisca la funzione:

```
Pietanza * leggiPietanze(char * nomeFile, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente l'elenco delle pietanze, legga da tale file le informazioni e le restituisca tramite un vettore di strutture dati di tipo **`Pietanza`** allocato dinamicamente della dimensione minima necessaria. Tramite l'intero **`dim`** passato per riferimento la funzione deve restituire la dimensione del vettore allocato. In caso di errore nell'apertura del file, la funzione deve restituire un puntatore a NULL, e **`dim`** pari a zero;

Si definisca la procedura:

```
void stampaPietanze(Pietanza * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Pietanza`**, e la sua dimensione **`dim`**, 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.

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

## Prova d'Esame 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

### *Esercizio 2 – Struttura dati Alunno, lettura e ordinamento degli alunni (modulo mensa.h/.c)*

Si definisca la struttura dati **`Alunno`** (nel file "mensa.h"), al fine di rappresentare i dati relativi al singolo alunno, cioè l'identificatore e l'elenco di tali ingredienti. In particolare, l'elenco degli ingredienti a cui l'alunno è intollerante deve essere memorizzato nella struttura **`Alunni`** tramite una lista di strutture dati di tipo **`Ingrediente`**.

Il candidato definisca una funzione:

```
Alunno * leggiAlunni(char * nomeFile, int * dim);
```

che, ricevuto in ingresso il nome di un file contente i dati relativi agli Alunni, e un intero **`dim`** passato per riferimento, restituisca un vettore allocato dinamicamente della dimensione minima, contenente i dati presenti nel file. Tramite l'intero **`dim`** passato per riferimento la funzione dovrà restituire la dimensione dell'array. Qualora vi siano errori nell'apertura del file specificato il puntatore restituito dovrà valere NULL, mentre la dimensione dovrà essere pari a zero.

Il candidato definisca poi una procedura:

```
void stampaAlunni(Alunno * vett, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Alunno`**, e la sua dimensione **`dim`**, ne stampi a video il contenuto.

Il candidato definisca poi una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Alunno`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in maniera decrescente in base al numero di ingredienti a cui l'alunno è intollerante; a parità numero di intolleranze, in ordine crescente in base all'identificatore dell'alunno. A tal fine, il candidato utilizzi uno qualunque tra gli 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.

### *Esercizio 3 – Verifica dei possibili menù (modulo mensa.h/mensa.c)*

L'azienda che produce i pasti deve fornire menù differenziati a seconda delle varie intolleranze. I dirigenti però hanno il timore che ciò non sia sempre possibile. Si tenga presente che ogni menù è composto da un primo e da un secondo, e che quindi entrambe le pietanze (primo e secondo) devono soddisfare i requisiti di intolleranza.

Si definisca una funzione:

```
Alunno * genera(Pietanza * p, int dimP, Alunno * a, int dimA, int * dimR);
```

che, ricevuto un ingresso un vettore di strutture dati di tipo **`Pietanza`** e la sua dimensione, e un vettore di strutture dati di tipo **`Alunno`** e la sua dimensione, restituisca un vettore allocato dinamicamente di strutture dati di tipo **`Alunno`** (non necessariamente della dimensione minima). Tramite il parametro **`dimR`** la funzione deve restituire la dimensione del vettore risultato. Il vettore risultato dovrà contenere la lista degli alunni per i quali non è possibile generare alcun menù.

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

Il candidato realizzi nella funzione **`main(…)`** un programma che, usando le informazioni fornite tramite il file di esempio fornito nello StartKit e le funzioni definite agli esercizi precedenti, stampi a video l'elenco degli alunni per i quali non è possibile creare un menù, cioè degli alunni per i quali non è possibile trovare una pietanza che possa fare da primo e una pietanza che possa fare da secondo.

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, 2014/2015 – Modulo 2

## Prova d'Esame 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

"element.h":

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

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_NOME 255


typedef struct {
	char nome[DIM_NOME];
} Ingrediente;

typedef struct {
	char nome[DIM_NOME];
	int tipo;
	Ingrediente * elenco;
	int size;
} Pietanza;

typedef Ingrediente element;




#endif
```

"element.c":

Non c'è in questo compito…

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

## Prova d'Esame 2A di Martedì 29 Gennaio 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 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 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

"mensa.h":

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

#ifndef _PIETANZE_H
#define _PIETANZE_H



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

typedef struct {
	int id;
	list elenco;
} Alunno;


// Es. 1
Pietanza * leggiPietanze(char * nomeFile, int * dim);
void stampaPietanze(Pietanza * v, int dim);

// Es. 2
Alunno * leggiAlunni(char * fp, int * dim);
void stampaAlunni(Alunno * vett, int dim);
void ordina(Alunno * v, int dim);

// Es. 3
Alunno * genera(Pietanza * p, int dimP, Alunno * a, int dimA, int * dimR);

#endif
```

"mensa.c":

```
#include "mensa.h"


// Es. 1

Pietanza * leggiPietanze(char * nomeFile, int * dim) {
	FILE * fp;
	Pietanza * result = NULL;
	Pietanza temp;
	int i;
	int j;

	*dim = 0;

	fp = fopen(nomeFile, "rt");
	if (fp != NULL) {
		while (fscanf(fp, "%s %d %d", temp.nome, &(temp.tipo), &(temp.size)) == 3) {
			for (i=0; i<temp.size; i++)
				fscanf(fp, "%*s");
```

```
                (*dim)++;
            }

            rewind(fp);
            result = (Pietanza*) malloc(sizeof(Pietanza) * *dim);
            for (i=0; i<*dim; i++) {
                fscanf(fp, "%s %d %d", temp.nome, &(temp.tipo), &(temp.size));
                temp.elenco = (Ingrediente*) malloc(sizeof(Ingrediente) * temp.size);
                for (j=0; j<temp.size; j++)
                    fscanf(fp, "%s", temp.elenco[j].nome);
                result[i] = temp;
            }
            fclose(fp);
        }
        return result;
}

void stampaPietanze(Pietanza * v, int dim) {
        int i;
        int j;

        for (i=0; i<dim; i++) {
            printf("%s, %d\n", v[i].nome, v[i].tipo);
            printf("Ingredienti:\n");
            for (j=0; j<v[i].size; j++)
                printf("\t%s\n", v[i].elenco[j].nome);
            printf("\n\n");
        }
}


// Es. 2
Alunno * leggiAlunni(char * nomeFile, int * dim) {
        FILE * fp;
        Alunno * result = NULL;
        Alunno temp;
        Ingrediente tempIng;
        int i;

        *dim = 0;
        fp = fopen(nomeFile, "rt");
        if (fp!=NULL) {
            fscanf(fp, "%d", dim);
            result = (Alunno*) malloc(sizeof(Alunno) *  *dim);
            for (i=0; i<*dim; i++) {
                fscanf(fp, "%d", &(temp.id));
                temp.elenco = emptylist();
                while (fgetc(fp) == ' ') {
                    fscanf(fp, "%s", tempIng.nome);
                    temp.elenco = cons(tempIng, temp.elenco);
                }
                result[i] = temp;
            }
            fclose(fp);
        }
        return result;
}
```

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

## Prova d'Esame 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

```
void stampaAlunni(Alunno * vett, int dim) {
	int i;
	list temp;
	for (i=0; i<dim; i++) {
		printf("Alunno: %d\nIntollerante a:\n", vett[i].id);
		temp = vett[i].elenco;
		while (!empty(temp)) {
			printf("\t%s\n", head(temp).nome);
			temp = tail(temp);
		}
	}
}

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

int length(list l) {
	if (empty(l))
		return 0;
	else
		return 1 + length(tail(l));
}

int compare(Alunno a, Alunno b) {
	int result;
	result = length(b.elenco) - length(a.elenco);
	if (result==0)
		result = a.id-b.id;
	return result;
}

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

// Es. 3
int compatibile(Pietanza p, Alunno a) {
	int trovato = 0;
	list temp = a.elenco;
	int i;
```

```
	Ingrediente ing;

	while (!empty(temp) && !trovato) {
		ing = head(temp);
		for (i=0; i<p.size && !trovato; i++)
			if (strcmp(p.elenco[i].nome, ing.nome) == 0)
				trovato = 1;
		temp = tail(temp);
	}
	return !trovato;
}

int esisteUnPrimoCompatibile(Pietanza * p, int dimP, Alunno a) {
	int trovato = 0;
	int i;
	for (i=0; i<dimP && !trovato; i++) {
		if (p[i].tipo == 1 && compatibile(p[i], a))
			trovato = 1;
	}

	return trovato;
}

int esisteUnSecondoCompatibile(Pietanza * p, int dimP, Alunno a) {
	int trovato = 0;
	int i;
	for (i=0; i<dimP && !trovato; i++) {
		if (p[i].tipo == 2 && compatibile(p[i], a))
			trovato = 1;
	}

	return trovato;
}

Alunno * genera(Pietanza * p, int dimP, Alunno * a, int dimA, int * dimR) {
	Alunno * result;
	int i;

	*dimR = 0;
	result = (Alunno *) malloc(sizeof(Alunno) * dimA);
	for (i=0; i<dimA; i++) {
		if (!esisteUnPrimoCompatibile(p, dimP, a[i]) ||
!esisteUnSecondoCompatibile(p, dimP, a[i])) {
			result[*dimR] = a[i];
			(*dimR)++;
		}
	}
	if (*dimR == 0) {
		free(result);
		result = NULL;
	}
	return result;
}
```

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

## Prova d'Esame 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "mensa.h"


int main() {

	// Es. 1
	{
		Pietanza * vett;
		int dim;
		int i;

		vett = leggiPietanze("pietanze.txt", &dim);
		stampaPietanze(vett, dim);
		for (i=0; i<dim; i++)
				free(vett[i].elenco);
		free(vett);
	}

	// Es. 2
	{
		Alunno * alunni;
		int dim;
		int i;

		alunni = leggiAlunni("alunni.txt", &dim);
		stampaAlunni(alunni, dim);
		ordina(alunni, dim);
		stampaAlunni(alunni, dim);
		for (i=0; i<dim; i++)
				freelist(alunni[i].elenco);
		free(alunni);
	}

	// Es. 3 & 4
	{
		Pietanza * piet;
		int dimP;
		Alunno * alunni;
		int dimA;
		int i;
		Alunno * r;
		int dimR;

		piet = leggiPietanze("pietanze.txt", &dimP);
		alunni = leggiAlunni("alunni.txt", &dimA);

		if (piet != NULL && alunni != NULL) {
			r = genera(piet, dimP, alunni, dimA, &dimR);
			ordina(r, dimR);
			printf("Alunni per i quali non e' possibile fare un menu:\n");
			stampaAlunni(r, dimR);
		}
```

```
            if (piet != NULL) {
                  for (i=0; i<dimP; i++)
                        free(piet[i].elenco);
                  free(piet);
            }
            if (alunni != NULL) {
                  for (i=0; i<dimA; i++)
                        freelist(alunni[i].elenco);
                  free(alunni);
            }
            // ATTENZIONE! STRUCTURE SHARIGN tra alunni e r...
            free(r);
      }

      return 0;
}
```

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

## Prova d'Esame 2A di Martedì 29 Gennaio 2015 – tempo a disposizione 2h

"pietanze.txt":

```
PastaAlPomodoro 1 5 pasta glutine pomodoro olio formaggio
PastaAlFormaggio 1 3 pasta glutine formaggio
PastaAlRagu 1 4 pasta glutine carneDiManzo carneDiMaiale
PastaInBianco 1 2 pastaSenzaGlutine burro
FormaggioGrigliato 2 1 formaggio
Bistecca 2 3 carneDiManzo olio sale
SalsicciaAiFerri 2 2 carneDiMaiale sale
uovaInCamicia 2 5 uova burro sale pepe olio
```

"alunni.txt":

5

11 glutine

17 formaggio

23 glutine formaggio olio

29 formaggio sale uova

31 glutine sale