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

## Prova d'Esame 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

**Prima di cominciare:** si scarichi dal sito **`http://esamix.labx`** il file **`StartKit2A.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"*.

In Italia ogni farmaco ha un nome, e contiene un determinato principio attivo in una certa quantità per dose di farmaco. Farmaci che hanno nomi diversi, ma che contengono lo stesso principio attivo nella stessa quantità sono detti *equivalenti*.
Una farmacia registra in un file di testo di nome "**`magazzino.txt`**" l'elenco dei farmaci a disposizione per la vendita: in particolare in ogni riga vengono memorizzati il **nome** del farmaco (una stringa di al più 255 caratteri, senza spazi), il nome del **principio** attivo (una stringa di al più 255 caratteri, senza spazi), la **quantità** in milligrammi di principio attivo (un intero), e infine il **numero** di confezioni di farmaco disponibili nella farmacia (un intero). Ogni campo è separato dal precedente tramite uno spazio.
I clienti si presentano alla farmacia con una ricetta per acquistare i farmaci. Grazie all'informatizzazione, ogni ricetta è un file di testo contenente un elenco di *prescrizioni* dove ogni prescrizione riguarda un farmaco (indicato tramite il suo nome) e il numero di confezioni da acquistare. Un esempio di ricetta è il file "**`ricetta1.txt`**" presente nello StartKit. In ogni riga del file è presente il nome di un farmaco e a seguire, separati da uno spazio, il numero di scatole da acquistare (un intero): ogni riga quindi rappresenta una singola prescrizione, e l'insieme delle prescrizioni presenti in un file rappresenta una ricetta.

### *Esercizio 1 – Strutture dati Farmaco e funzioni di lettura/scrittura (mod. element.h/.c e magazzino.h/.c)*

Si definisca un'opportuna struttura dati **`Farmaco`**, al fine di rappresentare i dati relativi ad ogni farmaco disponibile in farmacia: in particolare si dovrà tenere traccia del nome del farmaco, del principio attivo, della quantità di principio attivo, e del numero di confezioni disponibili.
Si definisca la funzione:

```
Farmaco * leggiMagazzino(char* fileName, int *dim);
```

che, ricevuto in ingresso il nome di un file di testo contenente l'elenco dei farmaci disponibili in farmacia, restituisca un array di strutture dati di tipo **`Farmaco`** 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.
Si definisca la funzione:

```
void scriviMagazzino(char* fileName, Farmaco * v, int dim);
```

che, ricevuto in ingresso il nome di un file di testo, un vettore **`v`** di strutture dati di tipo **`Farmaco`** e la dimensione **`dim`** di tale vettore, scriva sul file di testo indicato come parametro l'elenco dei farmaci specificati nel vettore.
Si definisca una procedura:

```
int ordina(Farmaco * v, int dim);
```

che, ricevuto in ingresso un vettore **`v`** di strutture dati di tipo **`Farmaco`** rappresentante le disponibilità in farmacia, e la dimensione **`dim`** di tale vettore, ordini il vettore secondo il seguente criterio: gli elementi devono essere ordinati in senso lessicografico in base al principio attivo del farmaco; in caso di principio attivo uguale, allora in maniera decrescente in base alla quantità di principio attivo presente nel farmaco; in caso di uguale quantità, allora in senso lessicografico in base al nome del farmaco. **Il candidato, per effettuare l'ordinamento, usi l'algoritmo InsertSort visto a lezione.**
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 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

### *Esercizio 2 – Struttura dati Prescrizione e lettura/scrittura (moduli element.h/.c e magazzino.h/.c)*

Si definisca un'opportuna struttura dati **`Prescrizione`**, al fine di rappresentare ognuna delle prescrizioni contenute in una ricetta, ovvero nome del farmaco e numero di confezioni da acquistare.
Il candidato definisca una funzione:

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

che ricevuta il nome di un file contenente l'elenco di prescrizioni, ne legga il contenuto e restituisca una lista di strutture dati di tipo **`Prescrizione`**.
Il candidato definisca poi una funzione:

```
int disponibile(Farmaco * v, int dim, Prescrizione p);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Farmaco`** rappresentante la disponibilità della farmacia, e la dimensione di tale vettore, e una prescrizione **`p`**, determini se la farmacia ha in disponibilità il farmaco specificato nel numero di confezioni richieste. In particolare, la funzione restituisca un valore compreso tra zero e **`dim`** se il farmaco con lo stesso nome è disponibile nel numero di confezioni richieste, dove il valore restituito indica l'indice nell'array **`v`** dove si trova tale farmaco. La funzione restituisca "-1" se il farmaco con lo stesso nome è presente in magazzino, ma non ci sono confezioni a sufficienza; la funzione restituisca "-2" se il farmaco è assente nel magazzino.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

### *Esercizio 3 –Ricerca dei generici e verifica di una ricetta (modulo magazzino.h/magazzino.c)*

Il candidato definisca una funzione:

```
int trovaGenerico(Farmaco * v, int dim, Prescrizione p);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Farmaco`** e la sua dimensione (rappresentanti il magazzino della farmacia), ed una prescrizione **`p`**, determini se esiste un farmaco equivalente in magazzino, nel numero di confezioni richiesto. Si ricordi che un **`Farmaco f1`** è **equivalente** a un **`Farmaco f2`** se hanno lo stesso principio attivo presente nella stessa quantità. Si noti poi che la prescrizione **`p`** specifica il nome del farmaco e il numero di confezioni richieste: poiché la prescrizione non specifica né principio attivo né quantità, per determinare se esista un farmaco equivalente sarà necessario prima determinare principio attivo e quantità del farmaco specificato nella prescrizione. La funzione deve restituire l'indice a cui si trova il farmaco equivalente; deve restituire il valore "-1" , se esiste un farmaco equivalente ma non in numero di confezioni sufficiente; deve restituire "-2" se non esiste farmaco equivalente.
Il candidato definisca quindi una funzione:

```
int accettabile(Farmaco * v, int dim, list ricetta);
```

che, ricevuti in ingresso un vettore e la sua dimensione **`dim`** rappresentanti la disponibilità di magazzino della farmacia, e una lista rappresentante una ricetta (cioè una lista di strutture dati di tipo **`Prescrizione`**), restituisca un valore interpretabile come "vero" se tutte le prescrizioni nella lista possono essere soddisfatte (tramite i farmaci specificati o tramite farmaci equivalenti), "falso" altrimenti.

### *Esercizio 4 – Determinazione delle ricette accettabili, 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, stabilisca quali ricette siano soddisfacibili da parte della farmacia e quali no. A titolo di esempio, nello StartKit sono forniti due file rappresentanti due ricette diverse. "ricetta1.txt" è soddisfabile dalla farmacia, mentre "ricetta2.txt" non lo è.

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 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

"element.h":

```
#ifndef ELEMENT_H_
#define ELEMENT_H_

#include <string.h>

#define DIM_NOME 256
#define DIM_PRINCIPIO 256



typedef struct {
	char nome[DIM_NOME];
	char principio[DIM_PRINCIPIO];
	int quant;
	int disp;
} Farmaco;

typedef struct {
	char nome[DIM_NOME];
	int request;
} Prescrizione;
typedef Prescrizione element;

int compare(Farmaco f1, Farmaco f2);
int equivalente(Farmaco f1, Farmaco f2);

#endif /* ELEMENT_H_ */
```

"element.c":

```
#include "element.h"

int compare(Farmaco f1, Farmaco f2) {
	int result;
	result = strcmp(f1.principio, f2.principio);
	if (result == 0) {
		result = f2.quant - f1.quant;
		if (result == 0)
			result = strcmp(f1.nome, f2.nome);
	}
	return result;
}


int equivalente(Farmaco f1, Farmaco f2) {
	if (strcmp(f1.principio, f2.principio)== 0 && f1.quant == f2.quant)
		return 1;
	else
		return 0;
}
```

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

## Prova d'Esame 2A di Giovedì 31 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 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

"magazzino.h":

```
#ifndef MAGAZZINO_H_
#define MAGAZZINO_H_

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

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

Farmaco * leggiMagazzino(char* fileName, int *dim);
void scriviMagazzino(char* fileName, Farmaco * v, int dim);
void ordina(Farmaco * v, int dim);

list leggiRicetta(char * fileName);
int disponibile(Farmaco * v, int dim, Prescrizione p);

int trovaGenerico(Farmaco * v, int dim, Prescrizione p);
int accettabile(Farmaco * v, int dim, list ricetta);

#endif /* MAGAZZINO_H_ */
```

"magazzino.c":

```
#include "magazzino.h"

Farmaco * leggiMagazzino(char* fileName, int *dim) {
	FILE * fp;
	Farmaco * result;
	Farmaco temp;
	int count;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp==NULL) {
		printf("Errore durante l'apertura del file %s\n", fileName);
		getchar();
		exit(-1);
	}
	else {
		count = 0;
		while(fscanf(fp, "%s%s%d%d", temp.nome, temp.principio, &(temp.quant),
&(temp.disp)) == 4)
			count++;
		rewind(fp);

		result = (Farmaco*) malloc(sizeof(Farmaco) * count);
		while(fscanf(fp, "%s%s%d%d", temp.nome, temp.principio, &(temp.quant),
&(temp.disp)) == 4) {
			result[*dim] = temp;
			(*dim)++;
		}
```

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

void scriviMagazzino(char* fileName, Farmaco * v, int dim) {
		FILE * fp;
		int i;

		fp = fopen(fileName, "wt");
		if (fp==NULL) {
			printf("Errore durante l'apertura del file %s\n", fileName);
			getchar();
			exit(-1);
		}
		else {
			for (i=0; i<dim; i++)
				fprintf(fp, "%s %s %d %d\n", v[i].nome, v[i].principio, v[i].quant,
v[i].disp);
			fclose(fp);
			return;
		}
		return;
}

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

// bubble sort
void bubbleSort(Farmaco * 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 insOrd(Farmaco v[], int pos){
		int i = pos-1, x = v[pos];
		while (i>=0 && compare(x,v[i])<0) {
			v[i+1] = v[i];
			i--;
		}
		v[i+1]=x;
}

void insertSort(Farmaco v[], int n){
		int k; for (k=1; k<n; k++)
		insOrd(v,k);
```

```
}

void quickSortR(Farmaco a[], int iniz, int fine) {
	int i, j, iPivot, pivot;
	if (iniz < fine) {
		i = iniz;
		j = fine;
		iPivot = fine;
		pivot = a[iPivot];
		do {
			while (i < j && compare(a[i], pivot)<=0) i++;
			while (j > i && compare(a[j], pivot)>=0) j--;
			if (i < j) scambia(&a[i], &a[j]);
		} while (i < j);
		if (i != iPivot && compare(a[i], a[iPivot])!=0) {
			scambia(&a[i], &a[iPivot]);
			iPivot = i;
		}
		if (iniz < iPivot - 1)
			quickSortR(a, iniz, iPivot - 1);
		if (iPivot + 1 < fine)
			quickSortR(a, iPivot + 1, fine);
	}
}

void quickSort(Farmaco a[], int dim) {
	quickSortR(a, 0, dim - 1);
}

void ordina(Farmaco * v, int n) {
	//bubbleSort(v, n);
	//insertSort(v,n);
	//quickSort(v,n);
}

list leggiRicetta(char * fileName) {
	list result;
	FILE * fp;
	Prescrizione temp;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell apertura del file %s\n", fileName);
		getchar();
		exit(-1);
	}
	else {
		while (fscanf(fp, "%s%d", temp.nome, &(temp.request)) == 2)
			result = cons(temp, result);
		fclose(fp);
		return result;
	}
}

int disponibile(Farmaco * v, int dim, Prescrizione p) {
	int i;
```

```
	int trovato;

	trovato = 0;
	i = 0;
	while (i<dim && !trovato) {
		if (strcmp(v[i].nome, p.nome) == 0)
			trovato = 1;
		else
			i++;
	}
	if (trovato == 1) {
		if (v[i].disp >= p.request)
			return i;
		else
			return -1;
	}
	return -2;
}

int trovaGenerico(Farmaco * v, int dim, Prescrizione p) {
	int i;
	int disp;
	int trovato;
	Farmaco campione;

	disp = disponibile(v, dim, p);
	if ( disp == -2)
		return disp;
	if (disp >= 0)
		return disp;

	if (disp == -1) {
		// cerca il farmaco col nome giusto
		trovato = 0;
		i = 0;
		while (i<dim && !trovato) {
			if (strcmp(v[i].nome, p.nome) == 0)
				trovato = 1;
			else
				i++;
		}
		campione = v[i];
		i= 0;
		trovato = 0;
		while (i<dim && !trovato) {
			if (equivalente(v[i], campione) && v[i].disp>=p.request)
				trovato = 1;
			else
				i++;
		}
		if (trovato)
			return i;
		else
			return -1;
	}
	return -3; //possibile caso di errore... qui non dovrebbe arrivare mai...
}
```

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

## Prova d'Esame 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

```
int accettabile(Farmaco * v, int dim, list ricetta) {
	int acc;
	int disp;
	Prescrizione p;

	acc = 1;

	while (!empty(ricetta) && acc) {
		p = head(ricetta);
		disp = disponibile(v, dim, p);
		if (disp == -1)
			disp = trovaGenerico(v, dim, p);
		if (disp < 0)
			acc = 0;
		ricetta = tail(ricetta);
	}
	return acc;
}
```

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

## Prova d'Esame 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

"main.c":

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

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

int main(void) {
	{	// Es. 1
		Farmaco * mag;
		int dim;

		mag = leggiMagazzino("magazzino.txt", &dim);
		ordina(mag, dim);
		scriviMagazzino("prova.txt", mag, dim);
		free(mag);
	}
	{	// Es. 2
		Farmaco * mag;
		int dim;
		list ricetta;
		list temp;
		Prescrizione p;
		int pres;

		mag = leggiMagazzino("magazzino.txt", &dim);
		ordina(mag, dim);
		scriviMagazzino("prova.txt", mag, dim);
		ricetta = leggiRicetta("ricetta1.txt");
		temp = ricetta;
		while (!empty(temp)) {
			p = head(temp);
			pres = disponibile(mag, dim, p);
			switch (pres) {
				case -1:
					printf("Il farmaco %s e' presente, ma non in numero
sufficiente\n", p.nome);
					break;
				case -2:
					printf("Il farmaco %s non e' presente\n", p.nome);
					break;
				default:
					printf("Il farmaco %s e' presente, all'indice %d\n",
p.nome, pres);
					break;
			}
			temp = tail(temp);
		}
		freelist(ricetta);
		free(mag);
	}

	{	// Es. 3 && 4
			Farmaco * mag;
			int dim;
```

```
                list ricetta1;
                list ricetta2;

                mag = leggiMagazzino("magazzino.txt", &dim);
                ordina(mag, dim);
                scriviMagazzino("prova.txt", mag, dim);
                ricetta1 = leggiRicetta("ricetta1.txt");
                ricetta2 = leggiRicetta("ricetta2.txt");

                if (accettabile(mag, dim, ricetta1))
                        printf("La ricetta 1 è accettabile\n");
                else
                        printf("La ricetta 1 non puo' essere soddisfatta\n");
                if (accettabile(mag, dim, ricetta2))
                        printf("La ricetta 2 è accettabile\n");
                else
                        printf("La ricetta 2 non puo' essere soddisfatta\n");

                freelist(ricetta1);
                freelist(ricetta2);
                free(mag);
        }


    getchar();
    return 0;
}
```

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

## Prova d'Esame 2A di Giovedì 31 Gennaio 2013 – tempo a disposizione 2h

"magazzino.txt":

```
Aspirina acido_salicilico 500 20
CardioAspirina acido_salicilico 100 30
Aspriinetta acido_salicilico 100 10
BuonaAspirinetta acido_salicilico 100 5
AspirinaAdulti acido_salicilico 500 3
Enapren enalapril 5 3
Enalapril enalapril 5 17
EnaprenAdulti enalapril 20 4
EnalaprilAdulti enalapril 20  17
```

"ricetta1.txt":

```
BuonaAspirinetta 12
EnaprenAdulti 2
```

"ricetta2.txt":

```
BuonaAspirinetta 12
EnaprenAdulti 2
Prozac 2
```