Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 21 Dicembre 2009
Compito A

Prima di cominciare: si scarichi il file **`StartKit7A.zip`** contenente i file di esempio.
Avvertenze per la consegna: nominare i file sorgenti come richiesto nel testo del compito, 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 sorgente** ed i file contenuti nello StartKit.
Rispettare le specifiche, in particolare inserire le funzioni nei file specificati fra parentesi dopo il nome della funzione. Chi non rispetta le specifiche sarà opportunamente penalizzato. **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non ragionevole" di errori di compilazione.
Consiglio: per verificare l'assenza di *warnings*, effettuare di tanto in tanto un *Rebuild All*.

La segreteria studenti di una facoltà tiene traccia dei voti degli studenti tramite un file di testo "**`elenco.txt`**". In ogni riga di tale file vengono memorizzati i seguenti dati (tutti relativi allo stesso studente):

- Matricola (al più 14 caratteri, senza spazi), seguito dauno spazio.
- Cognome (al più 63 cartteri, senza spazi) , seguito da uno spazio.
- Nome (al più 63 caratteri, senza spazi), seguito da uno spazio.
- Il numero di esami sostenuti (un intero) e, di seguito, tutti i voti degli esami. Tali voti sono tutti espressi tramite un intero, e sono separati tra di loro da uno spazio.

Ogni volta che uno studente passa un'esame, la segreteria aggiunge nel file di testo una nuova riga con aggiornato (aggiunto) l'esame relativo allo studente (e non cancella la riga precedente). Dopo un po' quindi, il contenuto del file ad esempio potrebbe essere:

```
2148000974787 Chesani Federico 5 30 28 26 30 19
2141000867548 Giannnelli Carlo 8 30 30 30 30 30 30 30 30
2148000974787 Chesani Federico 6 30 28 26 30 19 29
2148000974787 Chesani Federico 7 30 28 26 30 19 29 30
```

Non è noto a priori quante righe siano state memorizzate nel file, e non è noto neanche quanti studenti diversi siano effetivamente presenti nel file (chi ha dato molti esami compare più volte, chi non ne ha dati non compare affatto). E' noto però che per il corso di studi in questione gli esami sono in tutto 22.

Esercizio 1 - Rappresentazione e lettura dei dati relativi ad uno studente (studente.h/studente.c)

Si definisca un'opportuna struttura dati **studente**, al fine di rappresentare i dati registrati su ogni singola riga, come specificato sopra. Quindi si realizzi una funzione:

```
int leggiStudente(FILE * fp, studente * dest);
```

che, ricevuto in ingresso un puntatore ad un file già opportunamente aperto, legga i dati relativi ad uno studente memorizzati su una riga e li memorizzi nella struttura dest passata per riferimento. La funzione deve restituire un valore intero interpretabile come "vero" se la lettura è andata a buon fine, altrimenti un valore intero interpretabile come "falso".

Il candidato realizzi poi una funzione:

```
int uguali(studente s1, studente s2);
```

che, ricevuti come parametro di ingresso due strutture dati s1 e s2 di tipo **studente**, restituisca un valore intero interpretabile come "vero" se le strutture riguardano la stessa matricola, lo steso cognome e lo stesso nome. La funzione deve restituire un valore intero interpretabile come falso in caso contrario. Nello **`StartKit7A.zip`** il candidato troverà a disposizione un file di testo, "**`elenco.txt`**", contenente una lista di sutdenti di esempio.

Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni qui definite, al fine di verificarne la corretta implementazione: il candidato abbia cura, dopo aver letto i dati relativi agli studenti, di stampare a video i dati letti al fine di verificarne l'effettiva e corretta corrispondenza con quanto

memorizzato in "**elenco.txt**". Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 2 - Lettura di tutti gli studenti (studente.h/studente.c)

Il candidato realizzi una opportuna funzione:

```
studente * leggiTuttiStudenti(char * fileName, int * dim);
```

che provveda a leggere dal file (di nome fileName) tutti i dati relativi agli studenti, ripetizioni comprese. Tale funzione deve allocare dinamicamente lo spazio strettamente necessario, e per leggere i dati presenti su ogni riga dovrà fare utilizzo della funzione di cui al punto 1). Tramite il paramtero dim passato per riferimento, la funzione dovrà restituire il numero di righe lette dal file.

Il candidato realizzi poi una funzione:

```
int contaDiversi(studente * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo studente e la dimensione di tale vettore, restituisca in uscita il numero di studenti differenti memorizzati in tale vettore. A tal scopo, si utilizzi la funzione uguali(...) definita al punto 1).

Esercizio 3 - Eliminazione dei dati ripetuti (studente.h/studente.c)

Il candidato realizzi una funzione:

```
studente * eliminaRipetuti(studente * v, int dim, int * newDim);
```

che, ricevuto in ingresso un vettore v di strutture dati di tipo studente e la dimensione dim di tale vettore, restituisca in un nuovo vettore allocato dinamicamente gli elementi del vettore iniziale v senza ripetizioni. Ovviamente, siccome riguardo uno stesso studente vi sono più strutture dati memorizzate, sarà necessario copiare nel vettore risultato la struttura dati più recente, cioè quella in cui lo studente ha sostenuto più esami. A tal scopo, si noti che la segreteria aggiunge sempre in fondo al file le nuove informazioni, e di fatto il file risulta essere parzialmente ordinato: relativamente ad ogni studente, la riga contenente più esami è sempre quella "più in fondo" nel file (e quindi più in fondo nel vettore v).

Esercizio 4 - Main (main.c)

Il candidato realizzi un programma che legga dal file "elenco.txt" i dati relativi a tutti gli studenti e stampi a video, senza ripetizioni, l'elenco degli studenti ed il numero di esami sostenuti. Ovviametne, il programma deve stampare per ogni studente l'informazione "più aggiornata".

Il candidato abbia infine cura di deallocare (al termine del programma) tutte le strutture allocate dinamicamente.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 21 Dicembre 2009
Compito A

"studente.h":

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


#ifndef STUDENTE
#define STUDENTE

#define DIM_MAT 15
#define DIM_COGNOME 64
#define DIM_NOME 64
#define MAX_ESAMI 22

typedef struct {
       char mat[DIM_MAT];
       char cognome[DIM_COGNOME];
       char nome[DIM_NOME];
       int num_esami;
       int voti[MAX_ESAMI];
}
studente;


int leggiStudente(FILE * fp, studente * dest);
int uguali(studente s1, studente s2);
studente * leggiTuttiStudenti(char * fileName, int * dim);
int contaDiversi(studente * v, int dim);
studente * eliminaRipetuti(studente * v, int dim, int * newDim);


#endif
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 21 Dicembre 2009
Compito A

"studente.c":

```
#include "studente.h"


int leggiStudente(FILE * fp, studente * dest) {
	int result = 1;
	int i;

	if (fscanf(fp, "%s%s%s%d",
				dest->mat,
				dest->cognome,
				dest->nome,
				&(dest->num_esami)) != 4)
		result = 0;
	for (i=0; result && i< (dest->num_esami); i++) {
		if (fscanf(fp, "%d", dest->voti + i) != 1)
			result = 0;
	}
	return result;
}



int uguali(studente s1, studente s2) {
	if (   (strcmp(s1.mat, s2.mat)==0) &&
		(strcmp(s1.cognome, s2.cognome)==0) &&
		(strcmp(s1.nome, s2.nome)==0))
		return 1;
	else
		return 0;
}



studente * leggiTuttiStudenti(char * fileName, int * dim) {
	studente temp;
	studente * result;
	int i;
	FILE * fp;

	*dim = 0;

	if ((fp=fopen(fileName, "r")) ==  NULL) {
		printf("Errore nell'apertura di %s\n", fileName);
		exit(1);
	}

	while (leggiStudente(fp, &temp))
		(*dim)++;
	rewind(fp);
	result = (studente *) malloc(sizeof(studente)* (*dim));
	for (i=0; i< (*dim); i++)
		leggiStudente(fp, &(result[i]) );
	fclose(fp);
	return result;
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 21 Dicembre 2009
Compito A

```
int contaDiversi(studente * v, int dim) {
	int result = 0;
	int i, j;
	int trovato;

	for (i=0; i<dim; i++) {
		trovato = 0;
		for (j=i+1; j<dim && !trovato; j++) {
			if (uguali(v[i], v[j]))
				trovato = 1;
		}
		if (!trovato)
			result++;
	}
	return result;
}



studente * eliminaRipetuti(studente * v, int dim, int * newDim) {
	int i, j, k;
	int trovato;
	studente * result;

	k = 0;
	*newDim = contaDiversi(v, dim);
	result = (studente *) malloc(sizeof(studente) * (*newDim));
	for (i=0; i<dim; i++) {
		trovato = 0;
		for (j=i+1; j < dim && !trovato; j++) {
			if (uguali(v[i], v[j]))
				trovato = 1;
		}
		if (!trovato) {
			result[k] = v[i];
			k++;
		}
	}
	return result;
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 21 Dicembre 2009
Compito A

"main.c":

```
#include "studente.h"


int main() {
	studente * listaRip;
	studente * lista;
	int dimListaRip, dimLista;
	int i;

	listaRip = leggiTuttiStudenti("elenco.txt", &dimListaRip);
	lista = eliminaRipetuti(listaRip, dimListaRip, &dimLista);

	for (i=0; i<dimLista; i++)
		printf("%s %s %s %d\n",
		lista[i].mat,
		lista[i].cognome,
		lista[i].nome,
		lista[i].num_esami);
	free(listaRip);
	free(lista);

	system("PAUSE");

	return 0;
}
```