Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 16 Giugno 2009
Compito A

Prima di cominciare: si scarichi il file **`StartKit4A.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*.

Un certo Corso di Laurea gestisce la registrazione dei voti sostenuti dai propri studenti tramite un sistema informatico. Tale sistema registra i dati relativi agli studenti ed agli esami superati in alcuni file di testo. In particolare, in un file di testo "**`listaStud.txt`**" vengono registrati, in ogni riga, **nome** (al più 63 caratteri senza spazi), **cognome** (al più 63 caratteri senza spazi), e **matricola** (una stringa, di al più 63 caratteri senza spazi).
Poi, per ogni studente iscritto (ed elencato nel file "**`listaStud.txt`**"), il sistema informatizzato crea un file specifico dove memorizzare i dati: il file si chiama "*NomeCognome.txt*", dove a *Nome* si sostituisce il nome dello studente, e a *Cognome* il suo cognome. Ad esempio, per lo studente Federico Chesani il file si chiama "FedericoChesani.txt". All'interno di tale file, su ogni riga, viene memorizzato il **codice** di un esame (una stringa di al più 5 caratteri) e, separato da uno spazio, il **voto** ricevuto (un intero).
Obiettivo del programma è "visualizzare" le informazioni in base agli esami, e non in base agli studenti: attualmente i file sono organizzati dal punto di vista degli studenti, e non forniscono dati aggregati in base al codice dell'esame. In particolare, si vuole conoscere, per ogni esame presente nei file, il numero di studenti che l'ha sostenuto, ed il voto medio conseguito. Bisogna porre particolare attenzione però al fatto che a volte vengono registrati dei voti insufficienti (cioè minori di 18): tali voti non devono essere conteggiati ai fini né del calcolo di quanti studenti hanno sostenuto tale esame, né del calcolo della media. Inoltre, è noto che gli esami distinti nel corso di laurea in questione sono in tutto 29.

Esercizio 1 – Lettura degli studenti (studenti.h/studenti.c)
Si definisca una struttura dati studente, al fine di rappresentare i dati relativi ad uno studente (nome, cognome, matricola). Quindi si realizzi una funzione:

```
studente leggiUnoStudente(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad un file già opportunamente aperto, legga i dati relativi ad uno studente (cioè i dati memorizzati su una stessa riga) e li restituisca come struttura dati studente.
Il candidato realizzi poi una funzione:

```
studente * leggiTuttiGliStudenti(char * nomeFile, int * dim);
```

che, ricevuto come parametro di ingresso il nome di un file, provveda ad aprirlo e ne legga il contenuto. A tal fine, si usi la funzione **`leggiUnoStudente(…)`** appena definita. Si noti che non è noto a priori quanti studenti siano registrati nel file: sarà quindi necessario determinare quanti studenti vi siano memorizzati, e poi allocare memoria dinamicamente nella dimensione opportuna.
Nello **`StartKit4A.zip`** il candidato troverà a disposizione un file di testo, "listaStud.txt", contenente una lista di nomi di studenti 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 stmapare a video i dati letti al fine di verificarne la effettiva e corretta corrispondenza con quanto memorizzato in "listaStud.txt". Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 2 – Generazione del nome del file contente i voti (studenti.h/studenti.c)
Il candidato realizzi una funzione:

```
char * creaNomeFile(studente s);
```

che, ricevuto in ingresso una struttura dati studente, generi il nome del file corrispondente che conterrà gli esami ed i voti di quel certo studente. Si ricorda che per ogni studente, il file corrispondente si chiamerà "*NomeCognome.txt*", dove a *Nome* ed a *Cognome* si dovranno sostituire rispettivamente il nome ed il cognome dello studente.
Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni definite al fine di verificarne la corretta implementazione. Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 3 – Struttura dati riepilogoEame e ricerca (studenti.h/studenti.c)

Al fine di determinare le statistiche per ogni esame, si definisca una opportuna struttura dati denominata **riepilogoEsame**, che contenga il **codice** dell'esame, il **numero** di studenti che l'hanno sostenuto, e la **somma totale** di tutti i voti presi da questi studenti (tale somma verrà poi utilizzata per calcolare la media).
Il candidato realizzi quindi la funzione:

```
int trovaPos(riepilogoEsame * re, int dim, char * codice);
```

che ricevuto in ingresso un array di strutture dati riepilogo Esame con la sua dimensione dim, ed un codice identificativo di un certo esame, restituisca la posizione della struttura dati relativa a quel codice, se già presente nell'array, o -1 altrimenti.

Esercizio 4 – Aggiornamento delle strutture dati riepilogoEame (studenti.h/studenti.c)

Al fine di calcolare le medie, il candidato proceda nel modo seguente: dopo aver allocato (staticamente/dinamicamente) un array di strutture dati **riepilogoEsame** (si ricorda che gli esami in tutto sono 29), proceda a riempire e/o aggiornare tale array di strutture dati tramite la funzione seguente:

```
void aggiornaEsamiSostenuti(char nomeFile, riepilogoEsame * re, int *dim);
```

La funzione riceve in ingresso il nome di un file contenente i risultati di uno studente, e un array di strutture dati **riepilogoEsame** con la sua dimensione **dim** (inizialmente questa dimensione sarà pari 0, e poi crescerà man mano). La funzione deve leggere dal file ogni esame, e verificato che il voto sia sufficiente, provvedere ad aggiornare la struttura dati. Si possono verificare due casi, a) l'esame in questione è già presente nell'array (cioè una struttura dati **riepilogoEsame** è già stata memorizzata precedentemente in relazione a quel determinato codice d'esame): in tal caso si deve procedere all'aggiornamento dei dati relativi all'esame. Oppure, caso b), nell'array non è stato ancora memorizzata nessuna struttura dati relativa a quell'esame: in tal caso bisogna provvedere a registrare le informazioni nella prima posizione libera nell'array (l'ultima), e poi incrementare opportunamente la dimensione dell'array.
Al fine di determinare se un certo esame sia già stato memorizzato o no nell'array, si usi la funzione di cui al punto 3.

Esercizio 5 – Main (main.c)

Il candidato realizzi un programma **`main.c`** che provveda a leggere i nomi degli studenti nel file "listaStud.txt" e a memorizzarli tramite la funzione di cui al punto 1. Quindi il programma allochi spazio sufficiente per memorizzare un array di strutture dati **riepilogoEsame** (inizialmente vuoto), e tramite la funzione di cui al punto 4 legga, per ogni studente memorizzato nel file "listaStud.txt", gli esami ed i voti ricevuti, ed aggiorni l'array di riepilogo. Infine il main stampi a video l'elenco degli esami con codice dell'esame, numero di studenti che l'hanno sostenuto e medie.
Il candidato abbia cura di deallocare (al termine del programma) tutte le strutture allocate dinamicamente.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 16 Giugno 2009
Compito A

"studenti.h":

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


#ifndef STUDENTI
#define STUDENTI


#define MAX_DIM 63

typedef struct {
	char nome[MAX_DIM];
	char cognome[MAX_DIM];
	char matr[MAX_DIM];
}
studente;

typedef struct {
	char esame[6];
	int numStudenti;
	int sommaVoti;
}
riepilogoEsame;

#endif


studente leggiUnoStudente(FILE * fp);
void stampaUnoStudente(studente s);
studente * leggiTuttiGliStudenti(char * nomeFile, int * dim);

char * creaNomeFile(studente s);

int trovaPos(riepilogoEsame * re, int dim, char * codice);
void aggiornaEsamiSostenuti(char * nomeFile, riepilogoEsame * re, int *dim);
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 16 Giugno 2009
Compito A

"studenti.c":

```
#include "studenti.h"


studente leggiUnoStudente(FILE * fp) {
	studente result;
	fscanf(fp, "%s %s %s", result.nome, result.cognome, result.matr);
	return result;
}

void stampaUnoStudente(studente s) {
	printf("Nome: %s\n", s.nome);
	printf("Cognome: %s\n", s.cognome);
	printf("Matr.: %s\n", s.matr);
}

studente * leggiTuttiGliStudenti(char * nomeFile, int * dim) {
	studente * result = NULL;
	studente temp;
	FILE * fp;
	int i;

	*dim = 0;
	if ((fp=fopen(nomeFile, "r")) == NULL) {
		*dim = -1;
		return result;
	}
	while (fscanf(fp, "%s %s %s", temp.nome, temp.cognome, temp.matr) == 3)
		*dim = *dim + 1;
	rewind(fp);

	result = (studente *) malloc(sizeof(studente) * *dim);
	for (i=0; i<*dim; i++)
		result[i] = leggiUnoStudente(fp);
	fclose(fp);
	return result;
}


char * creaNomeFile(studente s) {
	int dim = 0;
	int i;
	char * result;

	i = 0;
	while (s.nome[i] != '\0') {
		i++;
		dim++;
	}

	i = 0;
	while (s.cognome[i] != '\0') {
		i++;
		dim++;
	}
	dim = dim + 4 + 1;
	result = (char *) malloc(sizeof(char) * dim);

	strcpy(result, s.nome);
	strcat(result, s.cognome);
	strcat(result, ".txt");

	return result;
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 16 Giugno 2009
Compito A

```
int trovaPos(riepilogoEsame * re, int dim, char * codice) {
	int pos = -1;
	int i;
	for (i=0; i<dim && pos<0; i++) {
		if (strcmp(re[i].esame, codice) == 0)
			pos = i;
	}
	return pos;
}


void aggiornaEsamiSostenuti(char * nomeFile, riepilogoEsame * re, int *dim) {
	FILE * fp;
	char codiceEsame[6];
	int voto;
	int pos;

	if ((fp = fopen(nomeFile, "r")) == NULL) {
		exit(1);
	}
	while (fscanf(fp, "%s %d", codiceEsame, & voto) == 2) {
		if (voto >= 18) {
			pos = trovaPos(re, *dim, codiceEsame);
			if (pos >=0) {
				re[pos].numStudenti++;
				re[pos].sommaVoti = re[pos].sommaVoti + voto;
			}
			else {
				strcpy(re[*dim].esame, codiceEsame);
				re[*dim].numStudenti = 1;
				re[*dim].sommaVoti = voto;
				*dim = *dim + 1;
			}
		}
	}
	fclose(fp);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 16 Giugno 2009
Compito A

"main.c":

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

#include "studenti.h"

int main(void)
{

	studente * lista;
	riepilogoEsame re[29];
	char * nomeFile;
	int dim_re = 0;
	int dim, i;

	lista = leggiTuttiGliStudenti("listaStud.txt", &dim);
	for (i=0; i<dim; i++)
		stampaUnoStudente(lista[i]);

	for (i=0; i<dim; i++)
		printf("%s\n", creaNomeFile(lista[i]));

	for (i=0; i<dim; i++) {
		nomeFile = creaNomeFile(lista[i]);
		aggiornaEsamiSostenuti(nomeFile, re, &dim_re);
		free(nomeFile);
	}

	for (i=0; i<dim_re; i++)
		printf("Esame: %s, Studenti: %d, Media: %f\n", re[i].esame,
re[i].numStudenti,
                                        ((float) re[i].sommaVoti)/re[i].numStudenti );

	free(lista);

	system("PAUSE");

	return (0);
}
```