Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

Prima di cominciare: si scarichi il file **`StartKit3A.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 piccolo comune ha installato un sistema di rilevamento con telecamera per fotografare le automobili che attraversano un incrocio pericolosissimo con il rosso. Il sistema fotografa l'auto, estrae dalla foto la targa del veicolo, e registra l'infrazione in un file binario. In particolare, in strutture apposite denominate **Multa**, il sistema registra la **targa** dell'auto (una stringa di 7 caratteri), il **giorno** in cui è stata scattata la foto (un intero rappresentante i giorni trascorsi a partire dal 1 Gennaio di quest'anno), e **l'importo** della multa (un float).
Ogni mese il comune deve inviare agli automobilisti indisciplinati la multa. Purtroppo alcuni automobilisti prendono, nello stesso mese, più multe. Il comune invia a questi un'unica contravvenzione, di importo pari alla somma di tutte le singole multe.
In un secondo file, di testo, la motorizzazione fornisce al comune un elenco dei proprietari di auto. Il file, di nome **`utenti.txt`**, è così organizzato: nella prima riga è presente un solo intero, che indica quanti utenti sono registrati nel file; nelle righe successive, in ogni riga, è registrata la **targa** dell'auto (una stringa di 7 caratteri), il **cognome** del proprietario (una stringa di al più 15 caratteri, senza spazi), e i **punti** ancora disponibili sulla patente (un intero).

Esercizio 1 – Lettura/scrittura delle multe (multa.h/multa.c)

Nello **`StartKit3A.zip`** il candidato troverà a disposizione due file, "crea.h" e "crea.c". Il file crea.h contiene la definizione della struttura dati **Multa**, da utilizzare in alcuni esercizi. Il file crea.c contiene una funzione **`void crea(char * filename)`** che crea un opportuno file binario chiamato col nome passato per argomento. Il candidato abbia cura di invocare opportunamente la funzione **`crea(…)`** al fine di generare un file binario contenente alcuni dati di esempio.

Il candidato definisca la seguente funzione:

```
Multa * leggiMulte(char * nomeFile, char * targa, int * dim);
```

Tale funzione riceve in ingresso il nome di un file, la targa di un'auto, e restituisce un array di multe relative alla targa specificata (la dimensione di tale array è restituita tramite il parametro dim passato per riferimento). La funzione deve aprire il file binario contenente le multe registrate dal sistema, e leggere dal file tutte le multe relative alla targa. Il file delle multe può contenere un numero imprecisato di multe relative alla stessa targa: quindi la funzione dovrà dapprima calcolare quante multe vi sono, poi allocare la memoria necessaria, e quindi procedere a leggere le multe.
Contestualmente, il candidato scriva nel main opportune istruzioni per invocare la funzione qui definita, al fine di verificarne la corretta implementazione: a tale scopo il candidato crei un file binario (nome a piacere) tramite la funzione **`crea(…)`** fornita nello **`StartKit3A`**. Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 2 – Struttura dati Persona: lettura e stampa (multa.h/multa.c)

Il candidato definisca nel file multa.h una opportuna struttura dati di nome **Persona**, atta a contenere i dati forniti dalla motorizzazione (targa, cognome e punti patente); la struttura dati dovrà anche essere predisposta per contenere l'elenco delle eventuali multe assegnate alla specifica targa.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

Il candidato definisca poi delle funzioni per (a) leggere una riga dal file di testo riguardo una persona, e trovare pure tutte le multe ad essa assegnate; (b) leggere dati e multe relative a tutte le persone che hanno preso almeno una multa; (c) stampare a video i dati relativi ad una persona ed alle sue multe. In dettaglio:

a) La funzione:

```
Persona leggiPersona(FILE * utentiFP, char * nomeFileMulte);
```

riceve in ingresso un puntatore a struttura FILE relativa ad un file già aperto contenente l'elenco degli utenti, ed il nome del file binario contenente le multe. La funzione legge i dati relativi ad una singola persona, e poi tramite la funzione `leggiMulte(…)` definita nell'esercizio 1 recupera tutte le multe relative a quella targa ed aggiorna/riempie opportunamente la struttura dati Persona da restituire.

b) La funzione:

```
Persona * leggiTuttiMultati(char * utenti, char * nomeFileMulte, int * dim);
```

riceve in ingresso il nome del file che contiene la lista di utenti, ed il nome del file contenente le multe. La funzione deve restituire un elenco degli utenti che hanno ricevuto almeno una multa. A tal fine, si utilizzi in maniera opportuna la funzione definita al punto b). L'elenco dei multati deve essere restituito in forma di un array, allocato dinamicamente, di dimensione eventualmente maggiore di quanto strettamente necessario. La dimensione dell'array deve essere restituita tramite il parametro dim passato per riferimento.

c) La funzione:

```
void stampaPersona(Persona p);
```

riceve in ingresso una struttura dati di tipo Persona, e deve stampare a video, oltre ai dati relativi alla persona, anche l'elenco delle multe ricevute ed il totale che deve pagare.

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 – Aggiornamento punti patente (multa.h/multa.c)

All'atto di comunicare la multa, il comune vuole anche comunicare che i punti della patente sono stati decurtati. Per ogni multa infatti vengono tolti 5 punti, fino a che non si raggiungono i 0 punti (non si possono avere punti in negativo).

Il candidato definisca una funzione:

```
void aggiorna(Persona * lista,int dim);
```

che, ricevuti in ingresso una lista di strutture dati di tipo Persona e la sua dimensione, modifichi tale lista, aggiornando i punti patente (ogni multa sono 5 punti in meno).

Esercizio 4 – Ordinamento delle multe (multa.h/multa.c)

Si realizzi una procedura che, ricevuto in ingresso un array di strutture dati Persona, ordini tale array in ordine di importo totale da pagare (dal maggiore al minore). In caso di importi identici, si ordini l'array per ordine alfabetico del cognome.

```
void ordina(Persona * p, int dim);
```

A tal scopo, si utilizzi uno qualunque degli algoritmi di ordinamento visti a lezione, avendo cura di modificarlo opportunamente per il caso in questione. Si consiglia di realizzare una funzione compare(…) che, ricevuti in ingresso due strutture Persona, restituisce -1, 0 o 1 a seconda che la prima struttura preceda, eguagli o segua la seconda struttura secondo il criterio specificato.

Esercizio 5 – Main (main.c)

Il candidato realizzi un programma `main.c` che chieda stampi a video l'elenco degli utenti multati, senza nessun ordinamento e senza aggiornare i punti patente. Quindi il main stampi l'elenco dei multati, ordinato secondo i criteri specificati nell'esercizio 4, con i punti della patente aggiornati opportunamente. Il candidato abbia cura di deallocare (al termine del programma) tutte le strutture allocate dinamicamente.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

"crea.h":

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

#ifndef MULTA
#define MULTA

typedef struct {
	char targa[8];
	int giorno;
	float importo;
} Multa;

#endif

void crea(char * filename);
```

"crea.c":

```
#include "crea.h"

void crea(char * filename) {
	FILE * fp;
	Multa m;
	int i;

	fp = fopen(filename, "wb");
	if (fp == NULL) {
		printf("Impossibile aprire il file %s\n", filename);
		system("PAUSE");
		exit(2);
	}
	for (i=0; i<5; i++) {
		if (i%2==1) {
			strcpy(m.targa, "AB123CF");
			m.giorno = i*10+5;
			m.importo = i*50.0F + 100;
		}
		else {
			strcpy(m.targa, "BA123ZX");
			m.giorno = i*10+5;
			m.importo = i*50.0F + 400;
		}
		fwrite(&m, sizeof(Multa), 1, fp);
	}

	fclose(fp);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

"multa.h":

```
#include "crea.h"
#include <string.h>

typedef struct {
	char targa[8];
	char cognome[16];
	int punti;
	Multa * lista;
	int dim;
} Persona;


Multa * leggiMulte(char * nomeFile, char * targa, int * dim);

Persona leggiPersona(FILE * utentiFP, char * nomeFileMulte);
Persona * leggiTuttiMultati(char * utenti, char * nomeFileMulte, int * dim);
void stampaPersona(Persona p);

void aggiorna(Persona * lista, int dim);
void ordina(Persona * p, int dim);
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

"multa.c":

```
#include "multa.h"

Multa * leggiMulte(char * nomeFile, char * targa, int * dim){
	FILE * fp;
	int count;
	Multa temp;
	Multa * result;

	fp = fopen(nomeFile, "rb");
	if (fp == NULL) {
		printf("Impossibile aprire il file %s in lettura.\n", nomeFile);
		system("PAUSE");
		exit(3);
	}

	count = 0;
	while (fread(&temp, sizeof(Multa), 1, fp) == 1) {
		if (strcmp(targa, temp.targa) == 0)
			count++;
	}
	rewind(fp);

	*dim = 0;
	result = (Multa *) malloc(sizeof(Multa) * count);
	while (fread(&temp, sizeof(Multa), 1, fp) == 1) {
		if (strcmp(targa, temp.targa) == 0) {
			result[*dim] = temp;
			*dim = *dim + 1;
		}
	}
	fclose(fp);
	return result;
}


Persona leggiPersona(FILE * utentiFP, char * nomeFileMulte) {
	Persona result;

	fscanf(utentiFP, "%s %s %d", result.targa, result.cognome, &(result.punti) );
	result.lista = leggiMulte(nomeFileMulte, result.targa, &(result.dim));
	return result;
}


Persona * leggiTuttiMultati(char * utenti, char * nomeFileMulte, int * dim) {
	Persona * result;
	Persona temp;
	FILE * fp;
	int count;
	int i;

	fp = fopen(utenti, "r");
	if (fp == NULL) {
		printf("Impossibile aprire il file %s\n", utenti);
		system("PAUSE");
		exit(3);
	}

	fscanf(fp, "%d", &count);
	result = (Persona *) malloc(sizeof(Persona) * count);
	*dim = 0;
	for (i=0; i<count; i++) {
		temp = leggiPersona(fp, nomeFileMulte);
		if (temp.dim > 0) {
```

```
                result[*dim] = temp;
                *dim = *dim + 1;
            }
        }
        fclose(fp);

        return result;
}

float calcolaTotale(Persona p) {
        int i;
        float tot = 0;
        for (i=0; i<p.dim; i++)
            tot = tot + p.lista[i].importo;
        return tot;
}

void stampaPersona(Persona p) {
        int i;
        float tot = 0;

        printf("%s %s, Punti: %d\n", p.cognome, p.targa, p.punti);
        for (i=0; i<p.dim; i++) {
            printf("Giorno: %d, Importo: %f\n", p.lista[i].giorno, p.lista[i].importo);
        }
        tot = calcolaTotale(p);
        printf("Totale da pagare: %f\n\n", tot);
}

void aggiorna(Persona * lista, int dim) {
        int i;

        for (i=0; i<dim; i++) {
            lista[i].punti = lista[i].punti - lista[i].dim * 5;
            if (lista[i].punti < 0)
                lista[i].punti = 0;
        }
}

void scambia(Persona * a, Persona * b) {
        Persona temp;
        temp = *a;
        *a = *b;
        *b = temp;
}



int compare(Persona p1, Persona p2) {
        if (calcolaTotale(p1) < calcolaTotale(p2))
            return 1;
        else
            if (calcolaTotale(p1) > calcolaTotale(p2))
                return -1;
            else
                return strcmp(p1.cognome, p2.cognome);
}

void bubbleSort(Persona v[], int n) {
        int i;
        int ordinato = 0;

        while (n>1 && ordinato==0) {
            ordinato = 1;
            for (i=0; i<n-1; i++) {
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

```
                    if ( compare(v[i],v[i+1])>0 ) {
                          scambia( &v[i], &v[i+1]);
                          ordinato = 0;
                    }
              }
              n--;
       }
}

void ordina(Persona * p, int dim) {
       bubbleSort(p, dim);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 11 Febbraio 2009
Compito A

"main.c":

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

#include "crea.h"
#include "multa.h"


int main(void)
{
	Persona * lista;
	int dim;
	int i;

	crea("multe.dat");
	lista = leggiTuttiMultati("utenti.txt", "multe.dat", &dim);
	for (i=0; i<dim; i++)
		stampaPersona(lista[i]);

	aggiorna(lista, dim);
	ordina(lista, dim);
	for (i=0; i<dim; i++)
		stampaPersona(lista[i]);

// e la deallocazione????
	system("PAUSE");

	return (0);
}
```