SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

Prima di cominciare: si scarichi il file **`StartKit0B.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*.

Una catena americana di pizzerie permette la prenotazione di pizze tramite il proprio sito Web. Si deve realizzare un programma che, forniti un insieme di ordini di pizze e il listino dei prezzi dei singoli ingredienti, calcoli la spesa da fatturare ad ogni cliente. Le informazioni sono contenute in due differenti file, **ingredienti.txt** e **ordini.bin**.

Il file di testo **ingredienti.txt** memorizza, in ogni riga:
- nome dell'ingrediente (al più 24 caratteri, senza spazi)
- prezzo di tale ingrediente (un float)

Il file binario **ordini.bin** memorizza invece gli ordini: all'inizio è salvato il numero di ordini presenti in tutto il file (int), e di seguito, sono registrate ***tutte le pizze ordinate***. Esattamente in questo ordine, per ogni pizza, sono salvati:
- nome del cliente che ha ordinato la pizza (al più 64 caratteri)
- numero di ingredienti presenti su quella pizza (al più 10)
- array di 10 elementi (non tutti usati) di ingredienti (composti dalla coppia nomeIngrediente-prezzo)

Ovviamente il file ***può contenere più pizze ordinate dalla stessa persona*** e le liste degli ingredienti ivi registrati, anche se composte dalla coppia nomeIngrediente-prezzo, in realtà riportano in modo uniforme solo il nome dell'ingrediente: infatti il prezzo è soggetto a frequenti modifiche (fa testo in tal senso il solo listino prezzi ufficiale dato nel file precedente).

Esercizio 1 - Lettura e scrittura degli ingredienti

Il candidato realizzi un modulo per la ***gestione del listino degli ingredienti***
In particolare, definisca una opportuna struttura dati **ingrediente**, per tenere traccia del nome di un ingrediente (al più 24 caratteri senza spazi) e del suo prezzo (double).
Il candidato poi definisca:
- insieme di primitive opportune per poter usare/gestire liste di strutture dati **ingrediente** (si faccia riferimento, a tal scopo, alle primitive sulle liste viste a lezione)
- funzione **leggiIngredienti(…)** che, ricevuto in ingresso il nome di un file, apra il file e restituisca una lista di strutture dati **ingrediente** lette da tale file
- funzione **scriviIngredienti(…)** che, ricevuto come parametri d'ingresso il nome di un file di testo e una lista di strutture dati **ingrediente**, scriva le informazioni relative agli ingredienti sul file indicato. In particolare, su ogni riga si scriva il nome dell'ingrediente e, separato da uno spazio, il suo prezzo

Nel main(), testare il funzionamento del modulo utilizzando il file di testo dato

Esercizio 2 - Modifica dei prezzi

Il candidato estenda il modulo definito al punto precedente realizzando:
- funzione **trovaPrezzo(…)** che, ricevuti in ingresso il nome di un ingrediente e una lista di strutture dati **ingrediente**, restituisca il prezzo di tale ingrediente

- funzione **aggiornaPrezzo(…)** che, ricevuti in ingresso la lista di strutture dati **ingrediente**, il nome di un ingrediente e un nuovo prezzo, restituisca la lista di ingredienti dove il prezzo dell'ingrediente specificato è stato aggiornato al nuovo valore. A tal scopo, il candidato consideri di accedere alla lista usando la notazione a puntatori

Nel main() il candidato modifichi la lista di ingredienti letti nell'esercizio precedente e controlli, salvando la lista aggiornata su un file, che tale modifica sia avvenuta con successo

Esercizio 3 - Gestione delle pizze

Il candidato realizzi un modulo per la ***gestione di una singola pizza.*** In particolare, si definisca una struttura dati **Pizza** contenente le seguenti informazioni (esattamente in questo ordine):

- nome del cliente che ha ordinato la pizza (al più 64 caratteri)
- numero di ingredienti (int)
- array di strutture dati **ingrediente**, di dimensione statica di 10 elementi

Tipicamente una pizza conterrà meno di 10 ingredienti: il numero di tali ingredienti è indicato nell'apposito campo.

Il candidato poi implementi una funzione **calcolaPrezzo(…)** che, ricevuti in ingresso una struttura dati di tipo **Pizza** e una lista di strutture dati **ingrediente**, calcoli il costo di tale pizza e lo restituisca come risultato della funzione.

Esercizio 4 - Gestione degli ordini

Il candidato realizzi un modulo di gestione degli ordini, che sono registrati su un file binario (a titolo di esempio, si consideri il file **ordini.bin** dato). Su tale file viene salvato innanzitutto il numero di pizze ordinate (int) e a seguire delle strutture dati di tipo **Pizza**.

Uno stesso cliente può ordinare più pizze, che possono essere registrate nel file anche in ordine non consecutivo; il file può contenere gli ordini relativi a diversi clienti. Il candidato implementi:

- funzione **leggiOrdini(…)** che, ricevuto in ingresso il nome di un file di ordini, apra tale file, legga il numero di pizze ivi registrate, allochi dinamicamente memoria sufficiente, e legga le strutture dati di tipo **Pizza** presenti in tale file. La funzione dovrà restituire un ***puntatore all'area di memoria allocata*** e, tramite un parametro passato per riferimento, il numero di pizze lette.
- funzione **sort(…)** che, utilizzando un algoritmo di ordinamento a scelta del candidato (tra quelli visti a lezione), ordini un array di strutture dati di tipo **Pizza** in base al nome del cliente che ha ordinato tale pizza.

Esercizio 5 - Calcolo della spesa dei clienti

Il candidato, utilizzando le funzioni definite negli esercizi precedenti, provveda a leggere il listino degli ingredienti dal file **ingredienti.txt**, aggiorni il prezzo dell'ingrediente "salame_piccante" a 2.00 euro e salvi (sovrascrivendo il file originale) la nuova lista di prezzi nello stesso file

Quindi il candidato legga dal file **ordini.bin** un array di pizze richieste da diversi clienti, ordini tale array in base al nome del cliente e calcoli per ogni cliente la spesa totale (data dalla somma del costo di ogni pizza richiesta dallo stesso cliente)

Suggerimento: si sfrutti il fatto che l'array è ordinato proprio in base al nome del cliente e che, quindi, pizze richieste dalla stessa persona risulteranno essere adiacenti

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"listino.h":

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

#ifndef LISTINO
#define LISTINO

#define MAX 25

typedef struct ingrediente
{
	char nome[MAX];
	float prezzo;
} Ingrediente;


typedef struct item_struct
{
	Ingrediente value;
	struct item_struct *next;
} item;

typedef item *list;



list leggiIngredienti(char *filename);
int scriviIngredienti(char *filename, list l);
float trovaPrezzo(char *nome, list l);
list aggiornaPrezzo(list l, char *nome, float nuovoPrezzo);

list emptyList();
int isEmpty(list l);
list cons(Ingrediente newValue, list l);
Ingrediente head(list l);
list tail(list l);


#endif
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"listino.c":

```
#include "listino.h"



list leggiIngredienti(char *filename)
{
	FILE * fp;
	char nome[MAX];
	float prezzo;
	Ingrediente temp;
	list result = emptyList();

	if ((fp = fopen(filename, "r")) == NULL)
	{
		return result;
	}

	while (fscanf(fp, "%s %f", nome, &prezzo) == 2)
	{
		strcpy(temp.nome, nome);
		temp.prezzo = prezzo;
		result = cons(temp, result);
	}

	fclose(fp);
	return result;
}

int scriviIngredienti(char *filename, list l)
{
	FILE * fp;

	if ((fp = fopen(filename, "w")) == NULL)
	{
		return -1;
	}

	while (!isEmpty(l))
	{
		fprintf(fp, "%s %f\n", head(l).nome, head(l).prezzo);
		l = tail(l);
	}

	fclose(fp);
	return 0;
}


float trovaPrezzo(char *nome, list l)
{
	while (!isEmpty(l))
	{
		if (strcmp(head(l).nome, nome) == 0)
		{
			return head(l).prezzo;
		}
		else
		{
			l = tail(l);
		}
	}
	return -1;
}
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

```
list aggiornaPrezzo(list l, char *nome, float nuovoPrezzo)
{
	list temp = l;
	int trovato = 0;
	while (!isEmpty(temp) && !trovato)
	{
		if (strcmp(head(temp).nome, nome) == 0)
		{
			trovato = 1;
			temp->value.prezzo = nuovoPrezzo;
		}
		else
		{
			temp = tail(temp);
		}
	}
	return l;
}



list emptyList()
{
	return NULL;
}

list cons(Ingrediente newValue, list l)
{
	item *temp = (item *)malloc(sizeof(item));
	temp->value = newValue;
	temp->next = l;
	return temp;
}

Ingrediente head(list l)
{
	return l->value;
}

list tail(list l)
{
	return l->next;
}

int isEmpty(list l)
{
	return l == NULL;
}
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"pizza.h":

```
#ifndef PIZZA
#define PIZZA

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

#include "listino.h"


typedef struct pizza
{
	char nomeCliente[65];
	int numeroIngredienti;
	Ingrediente topping[10];
} Pizza;


Pizza leggiPizza(FILE * fp);
float calcolaPrezzo(Pizza p, list l);
void scriviPizza();



#endif
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"pizza.c":

```
#include "pizza.h"



Pizza leggiPizza(FILE *fp)
{
	Pizza result;

	if (!feof(fp))
	{
		fread(&result, sizeof(Pizza), 1, fp);
	}
	return result;
}

float calcolaPrezzo(Pizza p, list l)
{
	int i;
	float result = 0;
	float temp;

	for (i = 0; i < p.numeroIngredienti; i++)
	{
		temp = trovaPrezzo(p.topping[i].nome, l);
		if (temp < 0)
		{
			return -1;
		}
		result = result + temp;
	}
	return result;
}


void scriviPizza()
{
	FILE *fp;
	Pizza temp;
	int num = 4;

	fp = fopen("ordini.bin", "wb");

	fwrite(&num, sizeof(int), 1, fp);

	strcpy(temp.nomeCliente, "Federico Chesani");
	strcpy(temp.topping[0].nome, "quattro_formaggi");
	strcpy(temp.topping[1].nome, "salame_piccante");
	temp.numeroIngredienti = 2;
	fwrite(&temp, sizeof(Pizza), 1, fp);

	strcpy(temp.nomeCliente, "Marco Montali");
	strcpy(temp.topping[0].nome, "margherita");
	strcpy(temp.topping[1].nome, "funghi");
	strcpy(temp.topping[2].nome, "prosciutto_cotto");
	temp.numeroIngredienti = 3;
	fwrite(&temp, sizeof(Pizza), 1, fp);

	strcpy(temp.nomeCliente, "Gabriele Zannoni");
	strcpy(temp.topping[0].nome, "margherita");
	strcpy(temp.topping[1].nome, "prosciutto_crudo");
	temp.numeroIngredienti = 2;
	fwrite(&temp, sizeof(Pizza), 1, fp);
```

```
        strcpy(temp.nomeCliente, "Federico Chesani");
        strcpy(temp.topping[0].nome, "margherita");
        strcpy(temp.topping[1].nome, "salsiccia");
        temp.numeroIngredienti = 2;
        fwrite(&temp, sizeof(Pizza), 1, fp);

        fclose(fp);
}
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"ordini.h":

```
#ifndef ORDINI
#define ORDINI

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

#include "pizza.h"

Pizza * leggiOrdini(char *filename, int *numPizze);
void naiveSortR(Pizza a[], int dim);



#endif
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"ordini.c":

```
#include "ordini.h"

Pizza* leggiOrdini(char *filename, int *numPizze)
{
	FILE *fp;
	Pizza *result;
	int temp;


	*numPizze = 0;
	if ((fp=fopen(filename, "rb")) == NULL)
	{
		return NULL;
	}

	fread(&temp, sizeof(int), 1, fp);
	result = (Pizza *) malloc(sizeof(Pizza) * temp);

	while (*numPizze < temp)
	{
		result[*numPizze] = leggiPizza(fp);
		*numPizze = *numPizze + 1;
	}

	fclose(fp);

	return result;
}


void swap(Pizza *a, Pizza *b)
{
	Pizza tmp = *a;
	*a = *b;
	*b = tmp;
}


void naiveSortR(Pizza a[], int dim)
{
	int i, posmin;
	Pizza min;
	if (dim == 1)
	{
		return;
	}
	for (posmin = 0, min = a[0], i = 1; i < dim; i++)
	{
		if (strcmp(a[i].nomeCliente,min.nomeCliente)<0)
		{
			posmin = i;
			min = a[i];
		}
	}
	if (posmin != 0)
	{
		swap(&a[0], &a[posmin]);
	}
	naiveSortR(&a[1], dim - 1);
}
```

SIMULAZIONE Fondamenti di Informatica T-1
Prova di Laboratorio - 14 Dicembre 2009
Compito B

"main.c":

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

#include "listino.h"
#include "pizza.h"
#include "ordini.h"


int main(void)
{
	list l;
	Pizza *p;
	int numPizze;
	int i;
	char *nome;
	float totale;

	scriviPizza();

	l = leggiIngredienti("ingredienti.txt");
	printf("Prezzo del salame piccante: %f\n", trovaPrezzo("salame_piccante", l) );
	l = aggiornaPrezzo(l, "salame_piccante", 2.00);
	printf("Prezzo del salame piccante: %f\n", trovaPrezzo("salame_piccante", l) );
	scriviIngredienti("ingredienti.txt", l);

	p = leggiOrdini("ordini.bin", &numPizze);
	naiveSortR(p, numPizze);
	for (i=0; i<numPizze; )
	{
		totale = calcolaPrezzo(p[i], l);
		nome = p[i].nomeCliente;
		i++;
		while (strcmp(nome, p[i].nomeCliente) == 0)
		{
			totale = totale + calcolaPrezzo(p[i], l);
			i++;
		}
		printf("%s deve pagare %f euro\n", nome, totale);
	}



	system("PAUSE");

	return (0);
}
```