Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 5 di Giovedì 15 Luglio 2010 – tempo a disposizione 2h30'

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit5.zip`** contenente i file necessari (*solution* di VS2008 e progetto compresi).
Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e **l'identificativo (A/B)** 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*, effettuare di tanto in tanto *"Rebuild All"*.

Una macchina automatica per la distribuzione di bevande ed altri alimenti è controllata da un programma che gestisce in memoria le giacenze dei vari prodotti, nonchè i codici identificativi e la loro descrizione. La macchina dispone di un numero (non noto a priori) di **slot**. In particolare, per ogni slot vengono memorizzate le seguenti informazioni:

- il **`codice`** dello slot, una stringa di al più 3 caratteri alfanumerici, che il cliente digita per selezionare il prodotto da acquistare (ad esempio, per l'acqua frizzante il codice potrebbe essere "B55");
- il **`prezzo`** del prodotto contenuto in quello slot (un float);
- la **`quantità`** di prodotti presenti in quel particolare slot (un intero);
- la **`descrizione`** del prodotto contenuto in quello slot, uan stringa di al più 63 caratteri, senza spazi;

Si noti che slot differenti possono contenere lo stesso prodotto: ad esempio ai codici "B55", "B56" e "B57" potrebbero corrispondere prodotti con la stessa descrizione "Acqua_Minerale_Gassata": questo accade perchè i prodotti più venduti spesso vengono inseriti più volte (in più slot) nella macchina distributrice.
Tutte le volte che la macchina viene spenta, salva i dati in un file binario ("dati.dat"), nel seguente modo: dapprima salva il numero di strutture dati **`Slot`**, e poi salva consecutivamente tutte le strutture dati di tipo **`Slot`**. All'accensione, la macchina legge da questo file binario i dati e li memorizza in memoria.
Si devono progettare alcune funzioni di utilità che verranno poi utilizzate durante il funzionamento della macchina.

Attenzione: ciascuna funzione realizzata deve essere opportunamente testata invocandola correttamente dal **`main(...)`** e mostrando sullo standard output il risultato dell'elaborazione effettuata. Inoltre tutta la memoria allocata dinamicamente (quindi anche gli elementi delle liste) deve essere opportunamente deallocata alla fine della computazione di ciascun esercizio.

Esercizio 1 – Strutture dati e funzioni di scrittura/lettura/stampa (moduli element.h e prodotti.h/prodotti.c)

a) Si definisca un'opportuna struttura dati **`Slot`** al fine di rappresentare il contenuto di uno slot, come descritto in precedenza;
b) Si realizzi la funzione

```
void scrivi(Slot* dati, int dim, char* nomeFile);
```

che, ricevuto il nome di un file (su cui scrivere in modo "binario"), ed un array di strutture dati di tipo **`Slot`** e la sua dimensione **`dim`**, scriva tali dati nel file indicato. In particolare, la funzione dapprima scriva il numero di strutture dati (il parametro **`dim`**) e poi il contenuto dell'array **`dati`**.
c) Si realizzi la funzione

```
Slot* leggi(char * fileName, int* dim);
```

che, ricevuto il nome di un file contenente un elenco di **`Slot`**, legga i dati contenuti e li memorizzi in un vettore di strutture dati di tipo **`Slot`** di dimensione opportuna. L'array restituito deve essere della dimensione esatta sufficiente a contenere tutti gli item registrati. Tale dimensione deve essere restituita tramite il parametro **`dim`**. Il file ha il formato descritto al punto precedente: all'inizio vi è memorizzato (in modo "binario") un intero che rappresenta il numero di strutture contenute nel file. Di seguito poi vi sono tutte le strutture dati, memorrizzate sempre in modo "binario".

d) Si realizzi poi la procedura

```
void stampaSlot(Slot *vector, int n);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Slot`**, stampi i primi **`n`** elementi di tipo **`Slot`** contenuti nel vettore.

Esercizio 2 – Ordinamento in base alla descrizione ed allo slot (element.h/prodotti.h/prodotti.c)

Al fine di conoscere i prodotti inseriti nella macchina e la loro posizione è necessario ordinare i dati letti dal file al momento dell'avvio (tramite le funzioni di cui all'Es. 1) in una opportuna struttura dati di tipo lista. Si definisca la funzione:

```
list ordina(Slot* FullVector, int dim);
```

che, ricevuto un vettore di **`Slot`** e la sua dimensione **`dim`**, restituisce una lista contente gli stessi dati, ordinati però dapprima in base alla descrizione e, nel caso di slot differenti ma con prodotti con la stessa descrizione, si utilizzi allora il codice identificativo dello **`Slot`**. Il criterio di ordinamento, sia per la descrizione che per il codice, deve essere quello lessicografico. Suggerimento: si definisca in "element.h/element.c" una funzione **`compare(...)`** ausiliaria per confrontare due strutture dati di tipo **`Slot`**.

Esercizio 3 – Conta dei prodotti diversi (prodotti.h/prodotti.c)

E' necessaria una funzione di utilià che conti quanti prodotti diversi (cioè con diversa descrizione) siano presenti nella macchina distributrice. Si definisca una funzione:

```
int conta(list elenco);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Slot`**, ordinate come specificato all'Es. 2, conti quanti prodotti sono presenti in tale lista.

Esercizio 4 – Calcolo delle giacenze (modulo element.h/element.c/prodotti.h/prodotti.c)

E' necessario calcolare le giacenze, al fine di poter riempire la macchina distributrice. Cioè per ogni tipo di prodotto (cioè con la stessa descrizione) si vuole sapere quanti oggetti sono ancora presenti nella macchina. A tal fine:

a) Si definisca una opportuna struttura dati **`totale`**, contenente due soli campi: una striga **`desc`**, ed un intero **`quantita`** (come i campi omonimi di cui all'esercizio 1).

b) Si realizzi la funzione

```
totale* calcolaTotali(list elenco, int * dimResult);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Slot`** (lista già ordianta come specificato allEs. 2), calcoli le giacenze totali dei prodotti con uguale descrizione, le memorizzi in strutture dati di tipo **`totale`** e restituisca tali informazioni tramite un array allocato dinamicamente, di dimensione **`dimResult`**. L'array restituito come risultato deve essere della dimensione fisica minima possibile.

Si abbia cura di testare la funzione nel main, stampando a video i totali così calcolati.

Dove specificato realizzare le funzioni nei moduli indicati; laddove non specificato si lascia libertà di scelta al candidato.

NOTA: Si abbia cura di deallocare la memoria allocata dinamicamente dal programma realizzato.

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 5 di Giovedì 15 Luglio 2010 – tempo a disposizione 2h30'

"element.h"

```
#ifndef _ELEMENT_H
#define     _ELEMENT_H
#define DIM_CODICE 4
#define DIM_DESC 64

typedef struct {
    char codice[DIM_CODICE];
      float price;
      int quantita;
      char desc[DIM_DESC];
} Slot;
typedef Slot element;

typedef struct {
      int quantita;
      char desc[DIM_DESC];
} totale;

int compare(element el1, element el2);

#endif      /* _ELEMENT_H */
```

"element.c"

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

int compare(element el1, element el2) {
      int result = 0;
      result = strcmp(el1.desc, el2.desc);
      if ( result == 0)
            result = strcmp(el1.codice, el2.codice);
      return result;
}
```

"prodotti.h"

```
#include "element.h"
#include "list.h"
#include <stdio.h>
#include <stdlib.h>

#ifndef PRODOTTI
#define PRODOTTI

void scrivi(Slot* dati, int dim, char* nomeFile);
Slot* leggi(char * fileName, int* dim);
void stampaSlot(Slot *vector, int n);

list ordina(Slot* FullVector, int dim);
int conta(list elenco);
totale* calcolaTotali(list elenco, int * dimResult);

#endif
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 5 di Giovedì 15 Luglio 2010 – tempo a disposizione 2h30'

"prodotti.c"

```
#include "prodotti.h"
#include "list.h"

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


void scrivi(Slot* dati, int dim, char* nomeFile) {
	FILE * fp;


	if ((fp=fopen(nomeFile, "wb")) == NULL) {
		printf("Problema nell'aprire il file: %s\n", nomeFile);
		system("pause");
		exit(-1);
	}
	fwrite(&dim, sizeof(int), 1, fp);
	fwrite(dati, sizeof(Slot), dim, fp);
	fclose(fp);
	return;
}

Slot* leggi(char * nomeFile, int* dim) {
	Slot * result;
	FILE * fp;

	if ((fp=fopen(nomeFile, "rb")) == NULL) {
		printf("Problema nell'aprire il file: %s\n", nomeFile);
		system("pause");
		exit(-1);
	}
	fread(dim, sizeof(int), 1, fp);
	result = (Slot*) malloc(sizeof(Slot) * *dim);
	fread(result, sizeof(Slot),*dim, fp);
	fclose(fp);
	return result;
}

void stampaSlot(Slot *v, int n) {
	int i;
	for (i=0; i<n; i++)
		printf("%s %f %d %s\n", v[i].codice, v[i].price, v[i].quantita,
v[i].desc);
	return;
}


list ordina(Slot* FullVector, int dim) {
	list result;
	int i;

	result = emptylist();
	for (i=0; i<dim; i++) {
		result = insord_p(FullVector[i], result);
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 5 di Giovedì 15 Luglio 2010 – tempo a disposizione 2h30'

```
int conta(list elenco) {
      Slot current;
      int result = 0;

      if (!empty(elenco)) {
            current = head(elenco);
            elenco = tail(elenco);
            result++;
            while (!empty(elenco)) {
                  if (strcmp(current.desc, head(elenco).desc) != 0) {
                        current = head(elenco);
                        result++;
                  }
                  elenco = tail(elenco);
            }
      }
      return result;
}



totale* calcolaTotali(list elenco, int * dimResult) {
      totale * result;
      Slot current;
      int i;

      *dimResult = conta(elenco);
      result = (totale *) malloc(sizeof(totale) * *dimResult);

      if (!empty(elenco)) {
            i=0;
            strcpy(result[i].desc, head(elenco).desc);
            result[i].quantita = head(elenco).quantita;
            elenco = tail(elenco);
            while (!empty(elenco)) {
                  if (strcmp(result[i].desc, head(elenco).desc) != 0) {
                        i++;
                        strcpy(result[i].desc, head(elenco).desc);
                        result[i].quantita = head(elenco).quantita;
                  }
                  else
                        result[i].quantita = result[i].quantita +
head(elenco).quantita;
                  elenco = tail(elenco);
            }
      }
      return result;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 5 di Giovedì 15 Luglio 2010 – tempo a disposizione 2h30'

"main.c"

```
#include "element.h"
#include "list.h"
#include "prodotti.h"
#include <stdio.h>
#include <stdlib.h>

int main() {
	/* ES. 1*/ {
		Slot * test;
		int dim;
		Slot content[4] = {
			{"B55",0.35F,6,"Acqua_Minerale_Frizzante"},
			{"B56",0.36F,6,"Acqua_Minerale_Frizzante"},
			{"B57",0.35F,8,"Acqua_Minerale_Naturale"},
			{"B58",0.35F,0,"Acqua_Minerale_Frizzante"}
		};
		printf("Esercizio 1:\n");
		scrivi(content, 4, "dati.dat");
		test=leggi("dati.dat", &dim);
		stampaSlot(test, dim);
		free(test);
		system("pause");
	}
	/* ES. 2*/ {
		Slot * test;
		list testList;
		int dim;
		printf("\n\nEsercizio 2:\n");
		test=leggi("dati.dat", &dim);
		stampaSlot(test, dim);
		printf("\n\n");
		testList = ordina(test, dim);
		showlist(testList);
		freelist(testList);
		free(test);
		system("pause");
	}
	/* ES. 3*/ {
		Slot * test;
		list testList;
		int dim;
		printf("\n\nEsercizio 3:\n");
		test=leggi("dati.dat", &dim);
		testList = ordina(test, dim);
		printf("Prodotti diversi: %d\n", conta(testList));
		freelist(testList);
		free(test);
		system("pause");
	}
	/* ES. 4*/{
		Slot * test;
		totale *  tot;
		list testList;
		int dim;
		int dimTot;
		int i;
		printf("\n\nEsercizio 4:\n");
		test=leggi("dati.dat", &dim);
		testList = ordina(test, dim);
		tot = calcolaTotali(testList, &dimTot);
		for (i=0; i<dimTot; i++)
			printf("%s %d\n", tot[i].desc, tot[i].quantita);
		freelist(testList);
		free(test);
		free(tot);
		system("pause");
	}
}
```