Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit4A.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"*.

Un negozio specializzato in macchine fotografiche vende sia oggetti nuovi che usati. Al fine di aumentare il numero di vendite, ha deciso di informatizzare l'elenco degli **`Item`** in vendita, per poi pubblicarlo in rete. Ha quindi proceduto ad analizzare le caratteristiche degli oggetti in vendita, giungendo alle seguenti conclusioni. Ogni **`Item`** è caratterizzato dalle seguenti informazioni:

- un campo **`id`**, codice identificativo unico dell'item (una stringa di 7 caratteri, senza spazi);
- un campo **`desc`**, una descrizione testuale dell'oggetto (una stringa di al più 255 caratteri, senza spazi);
- un campo **`price_base`**, rappresentante il prezzo all'ingrosso (o di acquisto dal mercato di seconda mano, se l'oggetto è usato) che il venditore ha pagato per l'item (un float);
- un campo **`price_sell`**, rappresentante il prezzo a cui il venditore mette in vendita l'oggetto (un float);
- un campo **`used`**, di tipo intero, che indica se l'oggetto è nuovo oppure usato: se vale "1", allora l'oggeto è usato, se vale "0" l'oggetto è nuovo.

Il padrone del negozio ha poi deciso di rifare l'inventario, e di salvare in un file di testo "**`inventario.txt`**" tutte le informazioni relative agli item posseduti. In ogni riga del file salva i dati relativi ad un singolo item: le informazioni salvate sono esattametne quelle di cui sopra, nell'ordine specificato, e separate tra di loro da uno spazio. Non è noto a priori quanti item siano memorizzati nel file. Si veda a titolo di esempio il file "**`inventario.txt`**" fornito nello start kit.

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. Si noti che alcune funzioni potrebbero richiedere la creazione di liste come strutture dati temporanee: si abbia cura di deallocare correttamente anche tali strutture.

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

a) Si definisca un'opportuna struttura dati **`Item`** al fine di rappresentare un singolo item, come descritto precedentemente;

b) Si realizzi la funzione

```
Item* readItem(char * fileName, int* dim);
```

che, ricevuto il nome di un file contenente un elenco di **`Item`**, legga i dati contenuti e li memorizzi in un vettore di strutture dati di tipo **`Item`** 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`**.

c) Si realizzi poi la procedura

```
void printItems(Item *vector, int n);
```

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

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

Esercizio 2 - Ricerca di items in base alla descrizione (modulo items.h/items.c)

Si definisca la funzione

```
list findItems(char* query, Item* vector, int dim);
```

che, dato un vettore di **Item** e la sua dimensione, restituisca una lista (ADT list) di **Item**, selezionati in base al seguente criterio: devono essere selezionati tutti gli item la cui descrizione comincia con quanto specificato in **query**. L'idea è che in **query** vengano specificati solo i primi caratteri della descrizione di un certo articolo, e che la funzione restituisca tutti gli articoli che effettivamente iniziano con tali caratteri. Si tenga presente che solitamente **query** è una stringa più corta di **desc**, ma potrebbe anche essere più lunga: in tal caso l'item non verifica il ciriterio di selezione. Ad esempio, facendo riferimento al file "**inventario.txt**", qualora la funzione sia invocata con **query="CanonEOS"**, viene restituita una lista di 4 elementi, tutti con descrizione "**CanonEOS550D**".

Esercizio 3 - Calcolo del "miglior business" (pedaggi.h/pedaggi.c)

Il proprietario del negozio vuole sapere qual'è l'item su cui realizzerà il miglior guadagno (in termini assoluti), a partire da una possibile descrizione dell'item. Si definisca la funzione

```
Item bestBusiness(Item* FullVector, int dim, char * query);
```

che, ricevuto un vettore di items, seleziona quelli che corrispondono al criterio **query** (a tal scopo si utilizzino le funzioni definite nell'esercizio 2). La funzione deve poi restituire l'item per cui il guadagno (**price_sell – price_base**) risulti essere maggiore. Qualora non esistesse nessun item corrispondente alla descrizione data tramite il parametro **query**, la funzione dovrà restituire un item di default col campo **desc** inizializzato alla stringa "**NO_ITEM_FOUND**". Ad esempio, facendo riferimento al file "**inventario.txt**", la funzione se invocata con parametro **query="CanonEOS"**, restituisce l'item con **id="0001237"**.

Esercizio 4 - Selezione ed Ordinamento in base alla descrizione ed al prezzo target (items.h/items.c/element.h)

Si definisca la funzione

```
Item * search(Item* fullVector, int dimFullVector, char * query, float
                                target_price, int *dim);
```

che, dato un array di strutture **Item** relativo a tutti gli articoli disponibili nel negozio (**fullVector** e relativa dimensione **dimFullVector**), restituisca un array di **Item** selezionati secondo un ben preciso criterio, e di dimensione **dim**. Tramite il parametro **query**, viene specificata la parte iniziale della descrizione degli item. Dovranno essere selezionati tutti gli item la cui descrizione comincia con **query**. A tal scopo, il candidato utilizzi la funzione di cui all'esercizio 2, che risolve esattamente questo sotto-problema. L'array inoltre dovrà essere ordinato nella maniera seguente: tramite il parametro **target_price**, viene specificato un prezzo di riferimento; all'inizio dell'array dovranno essere messi gli item il cui prezzo di vendita si avvicina "di più" al prezzo target (non importa se per eccesso o per difetto); in fondo all'array verranno messi gli item il cui prezzo di vendita si discosta maggiormente dal prezzo indicato tramite **target_price**.
Il candidato è libero di utilizzare una qualunque funzione di ordinamento tra quelle viste a lezione, tenendo presente di modificarle opportunamente.

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 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

"inventario.txt"

```
0001234 CanonEOS550D 500 580 1
0001235 CanonEOS550D 530 600 1
0001236 NikonD300 1200 1350 0
0001237 CanonEOS550D 600 730 0
0001238 CanonEOS550D 430 500 1
0001239 Canon1D 800 900 1
```

"element.h"

```
#ifndef _ELEMENT_H
#define     _ELEMENT_H

#define DIM_ID 8
#define DIM_DESC 256

typedef struct
{
    char id[DIM_ID];
      char desc[DIM_DESC];
      float price_base;
      float price_sell;
    int used;
} Item;

typedef Item element;


#endif      /* _ELEMENT_H */
```

"items.h"

```
#ifndef _ITEMS_H
#define     _ITEMS_H

#include "element.h"
#include "list.h"

Item* readItem(char * fileName, int* dim);
void printItems(Item *array, int n);

list findItems(char* query, Item* vector, int dim);

Item bestBusiness(Item* FullVector, int dim, char * query);

Item * search(Item* fullVector, int dimFullVector, char * query, float
target_price, int *dim);


#endif
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

"items.c"

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

Item* readItem(char * fileName, int* dim) {
	int count = 0;
	int i;
	FILE * fp;
	Item temp;
	Item * result;
	if ((fp=fopen(fileName, "r")) == NULL) {
		printf("A problem occurred when opening file %s.\n", fileName);
		system("PAUSE");
		exit(-1);
	}

	while (fscanf(fp, "%s %s %f %f %d",
		temp.id, temp.desc,
		&(temp.price_base), &(temp.price_sell), &(temp.used) )
				== 5)
		count++;
	rewind(fp);

	result = (Item *) malloc(sizeof(Item) * count);
	for (i=0; i<count; i++)
		fscanf(fp, "%s %s %f %f %d",
		result[i].id, result[i].desc, &(result[i].price_base),
		&(result[i].price_sell), &(result[i].used) );
	*dim = count;
	fclose(fp);
	return result;
}

void printItems(Item * vector, int n) {
	int i;
	for (i=0; i<n; i++) {
		printf("%s %s %f %f %d\n",
			vector[i].id, vector[i].desc,
			vector[i].price_base, vector[i].price_sell, vector[i].used );
	}
	return;
}

int matches(Item x, char * query) {
	int match = 1;
	char * desc = x.desc;
	while (*query != '\0' && match)
		if (*desc == '\0' || *query != *desc)
			match = 0;
		else {
			query++;
			desc++;
		}
	return match;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

```
list findItems(char* query, Item* vector, int dim) {
      list result;
      int i;

      result = emptylist();
      for (i=0; i<dim; i++) {
            if (matches(vector[i], query))
                  result = cons(vector[i], result);
      }
      return result;
}

Item bestBusiness(Item* fullVector, int dim, char * query) {
      list partialResult;
      list tempList;
      Item result;
      Item temp;
      int size = 0;

      partialResult = findItems(query, fullVector, dim);
      tempList = partialResult;
      if (!empty(tempList)) {
            result = head(tempList);
            tempList = tail(tempList);
            while (!empty(tempList)) {
                  temp = head(tempList);
                  if ( (temp.price_sell-temp.price_base) >
                              (result.price_sell-result.price_base))
                        result = temp;
                  tempList = tail(tempList);
            }
      }
      else {
            strcpy(result.desc, "NO_ITEM_FOUND");
      }
      freelist(partialResult);
      return result;
}

int compare(Item i1, Item i2, float target) {
      float diff1, diff2;
      diff1 = i1.price_sell-target;
      diff1 = (diff1>0)? diff1: -diff1;
      diff2 = i2.price_sell-target;
      diff2 = (diff2>0)? diff2: -diff2;
      if (diff1> diff2)
            return 1;
      else if (diff1<diff2)
            return -1;
      else
            return 0;
}

void insOrd(Item v[], int pos, float target) {
      int i = pos-1;
      Item x = v[pos];

      while (i>=0 && compare(x,v[i], target)<0) {
            v[i+1]= v[i];
```

```
            i--;
      }
      v[i+1]=x; /* inserisce l'elemento */
}

void insertSort(Item v[], int n, float target){
      int k;
      for (k=1; k<n; k++)
            insOrd(v,k, target);
}

Item * search(Item* fullVector, int dimFullVector, char * query, float
target_price, int *dim) {
      list partialResult;
      list temp;
      Item * result;
      int size = 0;
      int i;

      partialResult = findItems(query, fullVector, dimFullVector);
      temp = partialResult;
      while (!empty(temp)) {
            size++;
            temp = tail(temp);
      }
      result = (Item *) malloc(sizeof(Item) * size);
      // SOLUZIONE NON OTTIMA: Copio tutto in un array, e poi ordino
      temp = partialResult;
      for (i=0; i<size; i++) {
            result[i] = head(temp);
            temp = tail(temp);
      }
      insertSort(result, size, target_price);
      *dim = size;

      freelist(partialResult);
      return result;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 4a di Giovedì 10 Giugno 2010 – tempo a disposizione 2h30'

"main.c"

```
#include "element.h"
#include "list.h"
#include "items.h"

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


int main () {
	/* TEST ES 1 */
	{
		Item * inventario;
		int dim;
		printf("TEST 1\n");
		inventario = readItem("inventario.txt", &dim);
		printItems(inventario, dim);
		free(inventario);
	}
	/* TEST ES 2 */
	{
		Item * inventario;
		list selezione;
		int dim;
		printf("\n\nTEST 2\n");
		inventario = readItem("inventario.txt", &dim);
		selezione = findItems("CanonEOS", inventario, dim);
		showlist(selezione);
		free(inventario);
		freelist(selezione);
	}
	/* TEST ES 3 */
	{
		Item * inventario;
		Item best;
		int dim;
		printf("\n\nTEST 3\n");
		inventario = readItem("inventario.txt", &dim);
		best = bestBusiness(inventario, dim, "CanonEOS");
		printItems(&best, 1);
		free(inventario);
	}
	/* TEST ES 4 */
	{
		Item * inventario;
		Item * sel;
		int dim;
		int dimSel;
		printf("\n\nTEST 4\n");
		inventario = readItem("inventario.txt", &dim);
		sel = search(inventario, dim, "CanonEOS", 560, &dimSel);
		printItems(sel, dimSel);
		free(inventario);
		free(sel);
	}
	system("PAUSE");
	return 0;
}
```