Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit1B.zip`** contenente i file necessari (*solution* di VS2005/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 il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** e i file contenuti nello StartKit. Chi non rispetta le specifiche sarà opportunamente penalizzato.
Nota 1: **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non affrontabile" di errori di compilazione.
Nota 2: **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 videoteca gestisce il prestito dei CD in suo possesso tramite un sistema informatizzato. A tal scopo mantiene l'inventario dei CD che possiede e di quelli prestati o riconsegnati in due file distinti, rispettivamente **`cd_posseduti.txt`** e **`cd_prestati.txt`**.
Ciascuna riga del primo file contiene: codice CD (esattamente 8 caratteri), il carattere '@', il cognome del cantante (al più 20 caratteri, eventualmente contenente anche spazi), il carattere '@', il nome del cantante (al più 20 caratteri, eventualmente contenente anche spazi), il carattere '@', ed infine il numero di copie possedute di tale CD.
Ciascuna riga del secondo file contiene: +1 o -1 (un intero) a seconda che tale CD sia stato riconsegnato o prestato, uno spazio e un codice CD (esattamente 8 caratteri). Nel primo file ciascun codice CD compare una ed una sola volta, nel secondo file 0 o più volte (vedi file di esempio nello StartKit).

Esercizio 1 – Struttura dati CD *e funzioni di lettura (moduli element.h e cd.h/cd.c)*

Si definisca un'opportuna struttura dati **`CD`** che contenga codice CD, nome e cognome di un cantante, copie possedute e copie disponibili di un CD.
Si realizzi la funzione **`int calcolaCDPrestati(char* fileName, char* codice_cd);`** che, dato un codice CD ed un nome di file contenenti i CD prestati, restituisca il numero di copie attualmente prestate del CD con codice CD dato.

Esercizio 2 – Copie in possesso e copie disponibili (modulo cd.h/cd.c)

Si realizzi una funzione **`list disponibilita(char* filePosseduti, char* filePrestati);`** che, dati i nomi di due file contenenti i CD posseduti ed i CD prestati, crei la lista di strutture dati **`CD`** rappresentanti i Compact Disc in possesso alla videoteca, specificando opportunamente sia il numero di copie in possesso che il numero di copie disponibili di ciascun CD nella videoteca. A tal fine si utilizzi la funzione di cui all'esercizio 1.

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

Esercizio 3 – Comparazione fra due CD

Si realizzi una funzione **`compare(...);`** che, date due strutture dati di tipo **`CD`** C1 e C2, effettui il confronto fra queste in base al numero di copie attualmente prestate (differenza tra numero di copie possedute e numero di copie disponibili) e restituisca -1 se C1 è stato prestato meno di C2, +1 in caso contrario.
Se due CD sono stati prestati lo stesso numero di volte, bisogna controllare l'ordine lessicografico rispetto (rispettivamente) al cognome e al nome del cantante, restituendo -1 se l'autore di C1 precede lessicograficamente l'autore di C2, +1 altrimenti. Se anche cognome e nome sono identici, allora restituire 0.

Esercizio 4 – Lista ordinata di CD più prestati

Si realizzi una funzione **`piuPrestati(...);`** che, data una lista di CD, crei una nuova lista di CD avente in testa i CD più prestati ed in coda i CD meno prestati. Si consiglia l'utilizzo della funzione **`insord(...);`** vista a lezione per le liste modificandola opportunamente affinchè usi la funzione **`compare(...);`** realizzata precedentemente. Si suggerisce di prestare molta attenzione poichè tale funzione fornisce il criterio opposto di ordinamento, e quindi sarà necessario modificare opportunamente la funzione **`insord(...)`**.
La lista deve essere creata e gestita tramite le primitive definite per l'ADT lista.

Esercizio 5 – Selezione di CD per cantante

Si realizzi una funzione **`select(...);`** che, data una lista di CD e due stringhe ben formate **`inizio_cognome`** ed **`inizio_nome`**, restituisca un array contenente i CD della lista data il cui cantante abbia cognome che inizia per **`inizio_cognome`** ed il nome per **`inizio_nome`**. La funzione deve restituire opportunamente un riferimento all'array creato e la relativa dimensione. A tal scopo si realizzi una funzione **`iniziaPer(...);`** che date due stringhe (la prima più corta o della stessa lunghezza della seconda) restituisca 1 se i primi caratteri della seconda stringa sono esattamente i caratteri della prima, 0 altrimenti.
Il vettore restituito deve essere il più piccolo possibile.
La lista deve essere gestita tramite la notazione a puntatori e non tramite le primitive definite per l'ADT lista.

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

Ogni funzione realizzata deve essere opportunamente testata invocandola correttamente dalla funzione **`main()`**. Al fine di verificare la correttezza delle funzioni realizzate, si stampi a video l'elenco dei CD ordinati in base alla funzione di cui all'esercizio 4, il cui autore ha cognome con primi caratteri "Lu" e nome con primi caratteri "Ann".

Bonus: si abbia cura di deallocare la memoria allocata dinamicamente dal programma realizzato.

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

"element.h":

```
#ifndef _ELEMENT_H
#define	_ELEMENT_H


typedef struct{
	char codice[9];
	char cognome[21];
	char nome[21];
	int posseduti;
	int disponibili;
} CD;

typedef CD element;


#endif /* _ELEMENT_H */
```

"readField.h":

```
#include <stdio.h>

int readField(char buffer[], char sep, FILE *f);
```

"readField.c":

```
#include "readField.h"

int readField(char buffer[], char sep, FILE *f){
	int i = 0;
	char ch = fgetc(f);
	while (ch != sep && ch != 10 && ch != EOF){
		buffer[i] = ch;
		i++;
		ch = fgetc(f);
	}
	buffer[i] = '\0';
	return i;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

"list.h":

```
#ifndef LIST_H
#define LIST_H

#include "element.h"

typedef struct      list_element{
    element value;
    struct list_element  *next;
} item;

typedef  item* list;

typedef  int boolean;

/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);

#endif
```

"list.c":

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

#include "list.h"

/* OPERAZIONI PRIMITIVE */
list  emptylist(void)            /* costruttore lista vuota */
{
    return NULL;
}

boolean  empty(list l)     /* verifica se lista vuota */
{
    return (l==NULL);
}

list  cons(element e, list l)
{
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
  return(t);
}

element  head(list l) /* selettore testa lista */
{
  if (empty(l)) exit(-2);
  else return (l->value);
}

list  tail(list l)         /* selettore coda lista */
{
  if (empty(l)) exit(-1);
  else return (l->next);
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

"cd.h":

```
#ifndef _CD_H
#define      _CD_H

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

int calcolaCDPrestati(char* fileName, char* codice_cd);
list disponibilita(char* filePosseduti, char* filePrestati);
int compare(CD C1, CD C2);
list insord(element el, list l);
list piuLetti(list l);
int iniziaPer(char* str1, char* str2);
CD* select(char* inizio_cognome, char* inizio_nome, list l, int* dim);


#endif _CD_H
```

"cd.c":

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

#include "cd.h"
#include "readField.h"

int calcolaCDPrestati(char* fileName, char* codice_cd){
       FILE* fp;
       int ok=1, prestato, copiePrestate=0;
       char temp_codice[14];

       fp = fopen(fileName,"rt");
    if(fp==NULL) exit(-1);

       do{
              ok = ok && ( fscanf(fp, "%d %s", &prestato, temp_codice) == 2 );
              if( ok && (strcmp(temp_codice, codice_cd))==0 ){
                     copiePrestate -= prestato;
              }
       }while(ok);

       fclose(fp);
       return copiePrestate;
}

list disponibilita(char* filePosseduti, char* filePrestati){
       FILE* fp;
       list res;
       CD disco;
       int ok;

       fp = fopen(filePosseduti,"rt");
    if(fp==NULL) exit(-1);

       res = emptylist();

       do{
              ok = readField(disco.codice, '@', fp);
              ok = ok && readField(disco.cognome, '@', fp);
              ok = ok && readField(disco.nome, '@', fp);
              ok = ok && (fscanf(fp, "%d", &(disco.posseduti))==1 );
```

```
            disco.disponibili = disco.posseduti - calcolaCDPrestati(filePrestati,
disco.codice);
            if(ok){
                  res=cons(disco,res);
                  fgetc(fp);
            }
      }while(ok);

      fclose(fp);
      return res;
}

int compare(CD C1, CD C2){
      if( C1.posseduti-C1.disponibili > C2.posseduti-C2.disponibili){
            return +1;
      }
      else if( C1.posseduti-C1.disponibili < C2.posseduti-C2.disponibili){
            return -1;
      }
      else if(strcmp(C1.cognome, C2.cognome)>0){
            return +1;
      }
      else if(strcmp(C1.cognome, C2.cognome)<0){
            return -1;
      }
      else if(strcmp(C1.nome, C2.nome)>0){
            return +1;
      }
      else if(strcmp(C1.nome, C2.nome)<0){
            return -1;
      }
      else{
            return 0;
      }
}

list insord(element el, list l){
      if (empty(l)){
            return cons(el, l);
      }
      else if ( compare( el, head(l)) >= 0 ){
            return cons(el, l);
      }
      else{
            return cons(head(l), insord(el, tail(l)));
      }
}

list piuLetti(list l){
      list res;
      CD disco;

      res = emptylist();

      while(!empty(l)){
            disco = head(l);
            res = insord(disco, res);
            l = tail(l);
      }

      return res;
}

int iniziaPer(char* str1, char* str2){
      int res = 1;
      while(res==1 && *str1 != '\0'){
```

```
            if(*str1!=*str2){
                  res=0;
            }
            str1++;
            str2++;
      }
      return res;
}

CD* select(char* inizio_cognome, char* inizio_nome, list l, int* dim){
      CD disco;
      CD* res;
      list temp_list;

      *dim=0;
      temp_list=l;
      while(temp_list!=NULL){
            disco=temp_list->value;
            if( iniziaPer(inizio_cognome,disco.cognome) &&
iniziaPer(inizio_nome,disco.nome) ){
                  (*dim)++;
            }
            temp_list=temp_list->next;
      }

      res=(CD*)malloc(sizeof(CD)*(*dim));

      temp_list=l;
      *dim=0;
      while(temp_list!=NULL){
            disco=temp_list->value;
            if( iniziaPer(inizio_cognome,disco.cognome) &&
iniziaPer(inizio_nome,disco.nome) ){
                  res[*dim]=disco;
                  (*dim)++;
            }
            temp_list=temp_list->next;
      }

      return res;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 1B di Martedì 22 Dicembre 2009 – tempo a disposizione 2h30'

"main.c":

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

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

int main(void)
{
	list lista_dischi, lista_temp, lista_piuLetti;
	CD disco;
	CD* arrayDisco;
	int dimArray, i, copiePrestate;

	printf("Esercizio 1\n");
	copiePrestate = calcolaCDPrestati("cd_prestati.txt", "31231231");
	printf("copiePrestate %d\n", copiePrestate);
	printf("\n");

	printf("Esercizio 2\n");
	lista_dischi = disponibilita("cd_posseduti.txt", "cd_prestati.txt");
	lista_temp = lista_dischi;
	while(!empty(lista_temp)){
		disco = head(lista_temp);
		printf("%s %s %s %d %d\n", disco.codice, disco.cognome, disco.nome,
disco.posseduti, disco.disponibili);
		lista_temp = tail(lista_temp);
	}
	printf("\n");

	printf("Esercizio 3 e 4\n");
	lista_piuLetti = piuLetti(lista_dischi);
	lista_temp = lista_piuLetti;
	while(!empty(lista_temp)){
		disco = head(lista_temp);
		printf("%s %s %s %d %d\n", disco.codice, disco.cognome, disco.nome,
disco.posseduti, disco.disponibili);
		lista_temp = tail(lista_temp);
	}
	printf("\n");

	printf("Esercizio 5\n");
	arrayDisco = select("Lu", "Ann", lista_piuLetti, &dimArray);
	printf("dimArray %d\n", dimArray);
	for(i=0; i<dimArray; i++){
		printf("%s %s %s %d %d\n", arrayDisco[i].codice, arrayDisco[i].cognome,
arrayDisco[i].nome, arrayDisco[i].posseduti, arrayDisco[i].disponibili);
	}
	printf("\n");

	return (0);
}
```