Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit2A.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 il **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'azienda fornitrice di connessioni ad internet segna ogni giorno in un file di testo di nome "**`log.txt`**" gli utenti che si sono collegati tramite il sistema dell'azienda. In ogni riga del file di testo l'azienda segna l'**`id`** del cliente (una stringa di 7 caratteri alfanumerici, senza spazi), uno spazio di separazione, e la **`durata`** del collegamento in secondi (un intero). Ogni cliente si collega più volte, e per ogni collegamento è presente una riga nel file; le righe sono ovviamente ordinate cronologicamente in base all'istante in cui cliente si connette ad internet. Nello StartKit c'è a disposizione un file "**`log.txt`**" di esempio.

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

Si definisca un'opportuna struttura dati **`Collegamento`** al fine di rappresentare un collegamento da parte di un cliente. Si realizzi la funzione:

```
Collegamento * leggiTuttiCollegamenti(char* fileName, int * dim);
```

che, dato il nome di un file contenente i collegamenti, legga i dati ivi contenuti e li memorizzi in un vettore di strutture dati di tipo **`Collegamento`**. Non è noto a priori quanti dati siano memorizzati nel file; il vettore restituito dovrà comunque avere la dimensione minima necessaria. Tale dimensione dovrà essere restituita tramite il parametro **`dim`** passato per riferimento.

Si realizzi poi la procedura:

```
void stampaTuttiCollegamenti(Collegamento * v, int dim);
```

che, ricevuta in ingresso un vettore di strutture dati di tipo **`Collegamento`** e la sua dimensione **`dim`**, stampi a video i dati ivi contenuti. Si scriva poi nel **`main(...)`** un breve codice di prova delle due funzioni, usando a tal scopo il file "**`log.txt`**" contenuto nello StartKit; si abbia inoltre cura di deallocare eventuale memoria allocata dinamicamente. Una volta verificato il corretto funzionamento delle funzioni, non si cancelli il codice ma ci si limiti a commentarlo. A titolo di esempio, si prenda il file **`output.txt`** contenuto nello StartKit, che mostra un possibile output corretto.

Esercizio 2 - Ordinamento in base al cliente (modulo accessi.h/accessi.c)

Si definisca la funzione:

```
int compare(Collegamento c1, Collegamento c2);
```

che restituisca un valore minore di 0 se c1 "precede" **`c2`**, 0 se sono uguali o un valore maggiore di 0 se **`c1`** "segue" **`c2`**. Per il confronto tra le strutture dati, si usi il criterio lessicografico (ordine alfabetico) tra i campi **`id`**, e per strutture dati riguardanti lo stesso cliente, si confronti semplicemente le durate dei collegamenti.

Si definisca poi la funzione:

```
Collegamento * filtra(Collegamento * source, int dim, int * dimResult);
```

che, ricevuto in ingresso un vettore **`source`** di strutture dati di tipo **`Collegamento`** e la sua dimensione **`dim`**, restituisca un nuovo vettore allocato dinamicamente (della dimensione strettamente neccessaria **`dimResult`**). Il nuovo vettore dovrà contenere i valori del vettore **`source`**, tranne tutti i collegamenti che

hanno durata inferiore a 5 secondi: tali collegamenti sono infatti da considerarsi "errori" e quindi devono essere scartati.

Si definisca infine la funzione:
`Collegamento * ordinaSuCliente(Collegamento * source, int dim, int * dimResult);`
che, preso in ingresso un vettore **`source`** di strutture dati di tipo **`Collegamento`** e la sua dimensione **`dim`**, restituisca un nuovo vettore contenente gli stessi dati, ma filtrati e ordinati opportunamente. Per l'ordinamento, si utilizzi il criterio definito dalla funzione **`compare(...);`** (si usi quindi tale funzione). Per ordinare il vettore, il candidato può usare uno qualunque degli algoritmi di ordinamento visti a lezione. Invece per filtrare i dati, si usi a tal proposito la funzione **`filtra(...)`** definita precedentemente.
Si scriva poi nel main() un breve codice di prova delle funzioni, usando a tal scopo il file "**`log.txt`**" contenuto nello StartKit; si abbia inoltre cura di deallocare eventuale memoria allocata dinamicamente. Una volta verificato il corretto funzionamento delle funzioni, non si cancelli il codice ma ci si limiti a commentarlo. A titolo di esempio, si prenda il file **`output.txt`** contenuto nello StartKit, che mostra un possibile output corretto.

Esercizio 3 - Calcolo dei totali (accessi.h/accessi.c)

Si realizzi una funzione che, ricevuto in ingresso un vettore già filtrato ed ordinato di strutture dati di tipo **`Collegamenti`** e la sua dimensione, restituisca in uscita una lista di **`Collegamenti`** opportunamente elaborata. In particolare, la lista dovrà contenere una sola struttura dati di tipo **`Collegamento`** per ogni cliente. In tale struttura dovrà essere memorizzato, nel campo durata, il totale delle durate di tutti i collegamenti effettuati da quel particolare cliente.

Esercizio 4 - Calcolo e stampa dei totali (main.c)

Si realizzi un main che, usando il file di esempio **`log.txt`** fornito nello StartKit, stampi a video l'elenco dei clienti, e la durata totale dei collegamenti riferibili ad ogni cliente. A tal scopo si utilizzi la funzione di cui all'esercizio 3. Il programma abbia cura di deallocare eventuale memoria allocata precedentemente in maniera dinamica (siano vettori o liste).

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 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

"element.h":

```
#ifndef _ELEMENT_H
#define     _ELEMENT_H

#define DIM_ID 8

typedef struct
{
    char id[DIM_ID];
    int durata;
} Collegamento;

typedef Collegamento element;

#endif      /* _ELEMENT_H */
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 2A di Mercoledì 13 Gennaio 2010 – 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 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

"accessi.h":

```
#ifndef ACCESSI
#define ACCESSI

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

Collegamento * leggiTuttiCollegamenti(char* fileName, int * dim);
void stampaTuttiCollegamenti(Collegamento * v, int dim);

int compare(Collegamento c1, Collegamento c2);
Collegamento * filtra(Collegamento * source, int dim, int * dimResult);
Collegamento * ordinaSuCliente(Collegamento * source, int dim, int * dimResult);

list calcolaTotali(Collegamento * v, int dim);

#endif
```

"accessi.c":

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

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

Collegamento * leggiTuttiCollegamenti(char* fileName, int * dim) {
	FILE * fp;
	int count;
	int i;
	Collegamento * result;
	Collegamento temp;

	*dim = 0;
	count = 0;
	result = NULL;
	fp = fopen(fileName, "r");
	if (fp == NULL) {
		printf("Errore nell'aperturea del file: %s\n", fileName);
		return result;
	}

	while (fscanf(fp, "%s %d", temp.id, &(temp.durata)) == 2)
		count++;
	rewind(fp);
	result = (Collegamento *) malloc(sizeof(Collegamento) * count);
	for (i=0; i<count; i++) {
		fscanf(fp, "%s %d", result[i].id, &(result[i].durata));
		(*dim)++;
	}
	fclose(fp);
	return result;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

```
void stampaTuttiCollegamenti(Collegamento * v, int dim) {
      int i;
      for (i=0; i<dim; i++) {
            printf("ID: %s, Durata: %d\n", v[i].id, v[i].durata);
      }
}

int compare(Collegamento c1, Collegamento c2) {
      if (strcmp(c1.id, c2.id) !=0)
            return strcmp(c1.id, c2.id);
      else {
            if (c1.durata < c2.durata)
                  return -1;
            else
                  if (c1.durata > c2.durata)
                        return 1;
                  else
                        return 0;
      }
}

Collegamento * filtra(Collegamento * source, int dim, int * dimResult) {
      int count;
      int i;
      Collegamento * result;

      count = 0;
      *dimResult = 0;
      for (i=0; i<dim; i++) {
            if (source[i].durata >= 5)
                  count++;
      }
      result = (Collegamento *) malloc(sizeof(Collegamento) * count);
      for (i=0; i<dim; i++) {
            if (source[i].durata >= 5) {
                  result[*dimResult] = source[i];
                  (*dimResult)++;
            }
      }
      return result;
}

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

      while (i>=0 && compare(x,v[i])<0) {
            v[i+1]= v[i];
            i--;
      }
      v[i+1]=x; /* inserisce l'elemento */
}

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

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

```
Collegamento * ordinaSuCliente(Collegamento * source, int dim, int * dimResult) {
      Collegamento * filtrato;

      filtrato = filtra(source, dim, dimResult);
      insertSort(filtrato, *dimResult);
      return filtrato;
}

list calcolaTotali(Collegamento * v, int dim) {
      int i;
      list result;
      Collegamento temp;

      result = emptylist();

      if (dim>0) {
            temp = v[0];
            for (i=1; i<dim; i++) {
                  if (strcmp(temp.id, v[i].id) == 0)
                        temp.durata = temp.durata + v[i].durata;
                  else {
                        result = cons(temp, result);
                        temp = v[i];
                  }
            }
            result = cons(temp, result);
      }
      return result;
}
```

Fondamenti di Informatica T-1, 2009/2010 – Modulo 2
Prova d'Esame 2A di Mercoledì 13 Gennaio 2010 – tempo a disposizione 2h30'

"main.c":

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

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

int main() {


	/* TEST ES. 1 */
	{
		Collegamento * v;
		int dim;
		printf("*** Test Es. 1***\n");
		v = leggiTuttiCollegamenti("log.txt", &dim);
		stampaTuttiCollegamenti(v, dim);
		free(v);
		printf("*** Fine Es. 1***\n\n");
	}

	/* TEST ES. 2 */
	{
		Collegamento * v;
		Collegamento * v2;
		int dim;
		int dim2;
		printf("*** Test Es. 2***\n");
		v = leggiTuttiCollegamenti("log.txt", &dim);
		printf("Elenco non filtrato e non oridnato:\n");
		stampaTuttiCollegamenti(v, dim);
		v2 = ordinaSuCliente(v, dim, &dim2);
		printf("Elenco filtrato e oridnato:\n");
		stampaTuttiCollegamenti(v2, dim2);
		free(v);
		free(v2);
		printf("*** Fine Es. 2***\n\n");
	}

	/* TEST ES. 3-4 */
	{
		Collegamento * v;
		Collegamento * v2;
		list totali;
		list temp;
		int dim;
		int dim2;
		printf("*** Test Es. 3-4***\n");
		v = leggiTuttiCollegamenti("log.txt", &dim);
		v2 = ordinaSuCliente(v, dim, &dim2);
		totali = calcolaTotali(v2, dim2);
		temp = totali;
		while(!empty(temp)) {
			printf("ID: %s, Totale: %d\n", head(temp).id,
head(temp).durata);
			temp = tail(temp);
		}
```

```
            free(v);
            free(v2);
            temp = totali;
            while (!empty(temp)) {
                  totali = tail(temp);
                  free(temp);
                  temp = totali;
            }
            printf("*** Fine Es. 3-4***\n\n");
      }

      system("PAUSE");
      return 0;

}
```