Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit6A.zip`** contenente i file necessari (*progetto Visual Studio* ed eventuali altri file di esempio).
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.
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*, eseguire di tanto in tanto *"Rebuild All"*.

Un noto software implementa un sistema detto di "phoning home" per impedirne l'uso senza licenza. Ad ogni avvio, il programma invia ad un server, tramite una connessione internet, alcune informazioni: il server provvede a registrare su file tali informazioni, che verranno poi incrociate con gli archivi amministrativi dell'azienda per identificare possibili usi illeciti del software. In particolare, nel file di testo denominato "**`info.txt`**", il server memorizza le seguenti informazioni, separate da spazi, per ogni riga: numero **`seriale`** del software (una stringa di esattamente 15 caratteri alfanumerici, senza spazi); l'**`istante`** temporale (espresso in ore trascorse dal 1 Gennaio 2011, un intero); l'indirizzo **`IP`** del computer su cui il software sta eseguendo (una stringa di al più 15 caratteri alfanumerici). Si veda a titolo di esempio il file "**`info.txt`**" fornito nello StartKit. Non è noto a priori quante righe siano memorizzate nel file.
In un secondo file, di nome "**`archivio.txt`**", è contenuto l'elenco dei **`seriali`** software autorizzati, con il **`nome`** (stringa di al più 63 caratteri, senza spazi), e il **`cognome`** (ancora stringa di al più 63 caratteri, senza spazi): in ogni riga il numero seriale e, separati da spazi, nome e cognome.

Esercizio 1 – Strutture dati Record, Cliente e funzioni lettura/ordinamento (mod. element.h e info.h/info.c)

Si definisca un'opportuna struttura dati **`Record`**, al fine di rappresentare i dati relativi ad ogni registrazione, e memorizzati nel file info.txt, come specificato sopra. Si definisca poi una struttura dati **`Cliente`**, al fine di rappresentare i clienti ed il codice seriale del software in loro possesso, come descritto sopra.
Si definisca la funzione:

```
Record * leggiRecord(char* fileName, int *dim);
```

che, ricevuto in ingresso il nome di un file di testo contenente le informazioni, restituisca un array di strutture dati di tipo **`Record`** allocato dinamicamente (della dimensione minima necessaria), contenente tutte le informazioni presenti nel file indicato come parametro. Tramite il parametro **`dim`** la funzione deve restituire la dimensione del vettore.
Si definisca la funzione:

```
Cliente * leggiClienti(char* fileName, int *dim);
```

che, ricevuto in ingresso il nome di un file di testo contenente l'elenco dei clienti, restituisca un array di strutture dati di tipo **`Cliente`** allocato dinamicamente (della dimensione minima necessaria), contenente tutte le informazioni presenti nel file indicato come parametro. Tramite il parametro **`dim`** la funzione deve restituire la dimensione del vettore.
Il candidato definisca poi una procedura:

```
void ordina(Record * v, int dim);
```

che riceva in ingresso un vettore di strutture dati di tipo **`Record`** e la dimensione di tale vettore, e ordini il vettore secondo un ordine lessicografico sul codice seriale. A parità di seriale, le strutture dati devono essere ordinate in base all'istante temporale a cui fanno riferimento, in ordine crescente. Si usi a tal scopo un algoritmo di ordinamento a scelta tra quelli visti a lezione.

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

Esercizio 2 – Identificazione dei clienti attivi (modulo info.h/info.c)

Si realizzi una funzione:

```
list clientiAttivi(Record * r, int dimR, Cliente * C, int dimC);
```

che, ricevuta come parametro un vettore di strutture dati di tipo **`Record`** ordinato come specificato nell'esercizio 1 e la sua dimensione **`dimR`**, un vettore di strutture dati di tipo **`Cliente`** e la sua dimensione **`dimC`**, restituisca una lista dei clienti attivi. Un cliente è da considerarsi attivo se ha utilizzato almeno una volta il software. In altre parole, sono clienti attivi tutti coloro che possiedono il software (e che quindi compaiono nel file "archivio.txt"), e il cui seriale compaia almeno una volta nel file "info.txt".

Si realizzi una procedura ricorsiva:

```
void stampaListaClienti(list l);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Cliente`**, stampi a video il contenuto di tale lista. La procedura deve essere implementata in maniera ricorsiva.

Esercizio 3 –Identificazione degli illeciti (modulo info.h/info.c)

Si definisca poi una procedura:

```
list illeciti(Record * r, int dimR, Cliente * c, int dimC);
```

che, ricevuta come parametro un vettore di strutture dati di tipo **`Record`** ordinato come specificato nell'esercizio 1 e la sua dimensione **`dimR`**, un vettore di strutture dati di tipo **`Cliente`** e la sua dimensione **`dimC`**, restituisca una lista dei clienti il cui software è stato usato in maniera illecita. Un software è stato usato illecitamente se compaiono due registrazioni diverse (due Record diversi) relativi allo stesso seriale, ma usati da due indirizzi ip diversi, e in due istanti temporali distanti non più di 5 ore. La lista restituita deve essere ordinata in base al solo cognome del cliente, e non dovrà contenere ripetizioni. Ad esempio, usando i files forniti nello StartKit, la lista dovrà contenere (in ordine) le strutture dati "**`abcde98765abcde Federico Chesani`**" e "**`qwert54321qwert Carlo Giannelli`**".

Esercizio 4 – Stampa dei clienti attivi, dei clienti con possibili illeciti, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che, usando le informazioni fornite tramite i file di esempio forniti nello StartKit, e le funzioni definite agli esercizi precedenti, stampi a video la lista dei clienti attivi e di quelli sospetti di illeciti.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

"element.h":

```
#include <string.h>

#ifndef _ELEMENT_H
#define      _ELEMENT_H

#define DIM_SERIALE 16
#define DIM_NOME 64

typedef struct {
       char seriale[DIM_SERIALE];
       int time;
       char ip[16];
} Record;

typedef struct {
       char seriale[DIM_SERIALE];
       char nome[DIM_NOME];
       char cognome[DIM_NOME];
} Cliente;

typedef Cliente element;

int compare(Record r1, Record r2);

#endif /* _ELEMENT_H */
```

"element.c":

```
#include "element.h"

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


int compare(Record v1, Record v2) {
       int temp;

       temp = strcmp(v1.seriale, v2.seriale);
       if (temp == 0)
              temp = v1.time - v2.time;
       return temp;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

"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);
void freelist(list l);
int member(element el, list l);
list insord_p(element el, list l);

#endif
```

"list.c":

```
#include <stdio.h>
#include <stdlib.h>
#include <string.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);
}

int member(element el, list l) {
       if (empty(l))
              return 0;
       else {
              if (strcmp(el.seriale, head(l).seriale)==0)
                     return 1;
              else
                     return member(el, tail(l));
       }
}

void freelist(list l) {
       if (empty(l))
              return;
       else {
              freelist(tail(l));
              free(l);
       }
       return;
}

list insord_p(element el, list l) {
       list pprec, patt = l, paux;
       int trovato = 0;

       while (patt!=NULL && !trovato) {
              //if (compare(el, patt->value)<0)
              if (strcmp(el.cognome, patt->value.cognome)<0)
                     trovato = 1;
              else {
                     pprec = patt;
                     patt = patt->next;
              }
       }
       paux = (list) malloc(sizeof(item));
       paux->value = el;
       paux->next = patt;
       if (patt==l)
              return paux;
       else {
              pprec->next = paux;
              return l;
       }
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

"info.h":

```
#ifndef INFO
#define INFO

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

Record * leggiRecord(char* fileName, int *dim);
Cliente * leggiClienti(char* fileName, int *dim);
void ordina(Record v[], int n);

list clientiAttivi(Record * r, int dimR, Cliente * c, int dimC);
void stampaListaClienti(list l);

list illeciti(Record * r, int dimR, Cliente * c, int dimC);

#endif
```

"info.c":

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


Record * leggiRecord(char* fileName, int *dim) {
	FILE * fp;
	Record * result;
	Record temp;
	int i;

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file %s\n", fileName);
		system("pause");
		exit(-1);
	}
	else {
		*dim = 0;
		while (fscanf(fp, "%s %d %s", temp.seriale, &(temp.time), temp.ip)==3)
			(*dim)++;
		rewind(fp);
		result = (Record* ) malloc(sizeof(Record) * *dim);
		i=0;
		while (fscanf(fp, "%s %d %s", temp.seriale, &(temp.time), temp.ip)==3) {
			result[i] = temp;
			i++;
		}
		fclose(fp);
		return result;
	}
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

```
Cliente * leggiClienti(char* fileName, int *dim) {
	FILE * fp;
	Cliente * result;
	Cliente temp;
	int i;

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file %s\n", fileName);
		system("pause");
		exit(-1);
	}
	else {
		*dim = 0;
		while (fscanf(fp, "%s %s %s", temp.seriale, temp.nome, temp.cognome)==3)
			(*dim)++;
		rewind(fp);
		result = (Cliente *) malloc(sizeof(Cliente) * *dim);
		i=0;
		while (fscanf(fp, "%s %s %s", temp.seriale, temp.nome, temp.cognome)==3) {
			result[i] = temp;
			i++;
		}
		fclose(fp);
		return result;
	}
}


void scambia(Record *a, Record *b) {
	Record tmp = *a;
	*a = *b;
	*b = tmp;
}
// bubble sort
void ordina(Record v[], int n) {
	int i, ordinato = 0;
	while (n>1 && !ordinato) {
		ordinato = 1;
		for (i=0; i<n-1; i++)
			if (compare(v[i],v[i+1])>0) {
				scambia(&v[i],&v[i+1]);
				ordinato = 0;
			}
		n--;
	}
}


list clientiAttivi(Record * r, int dimR, Cliente * c, int dimC) {
	list result;
	int i;
	int j;
	int trovato;
	Cliente temp;

	result = emptylist();
```

```
	for (i=0; i<dimC; i++) {
		trovato = 0;
		for (j=0; j<dimR && !trovato; j++) {
			if (strcmp(c[i].seriale, r[j].seriale)==0)
				trovato = 1;
		}
		if (trovato && !member(c[i], result))
			result = cons(c[i], result);
	}
	return result;
}


void stampaListaClienti(list l) {
	if (empty(l))
		return;
	else {
		printf("%s %s %s\n", head(l).seriale, head(l).nome, head(l).cognome);
		stampaListaClienti(tail(l));
		return;
	}
}


list illeciti(Record * r, int dimR, Cliente * c, int dimC) {
	list result;
	int i;
	int j;
	int trovato;
	Cliente temp;

	result = emptylist();
	i=0;
	while (i<dimR-1) {
		trovato = 0;
		for (j=0; j<dimC && !trovato; j++) {
			if (strcmp(r[i].seriale, c[j].seriale)==0) {
				trovato = 1;
				temp = c[j];
			}
		}
		if (strcmp(r[i].seriale, r[i+1].seriale)==0 &&
			strcmp(r[i].ip, r[i+1].ip) !=0 &&
			(r[i+1].time-r[i].time)<5 &&
			! member(temp, result)
			)
				result = insord_p(temp, result);
		i++;
	}

	return result;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "info.h"

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


int main() {
	int i;
	int dimRecord;
	int dimClienti;
	Record * elenco;
	Cliente * elencoClienti;
	list cattivi;
	list attivi;

	printf("Lettura Record:\n");
	elenco = leggiRecord("info.txt", &dimRecord);
	for (i=0; i<dimRecord; i++)
		printf("%s %d %s\n", elenco[i].seriale, elenco[i].time, elenco[i].ip);
	printf("\n\n");

	printf("Ordinamento:\n");
	ordina(elenco, dimRecord);
	for (i=0; i<dimRecord; i++)
		printf("%s %d %s\n", elenco[i].seriale, elenco[i].time, elenco[i].ip);
	printf("\n\n");

	printf("Lettura Clienti:\n");
	elencoClienti = leggiClienti("archivio.txt", &dimClienti);
	for (i=0; i<dimClienti; i++)
		printf("%s %s %s\n", elencoClienti[i].seriale, elencoClienti[i].nome,
elencoClienti[i].cognome);
	printf("\n\n");

	attivi = clientiAttivi(elenco, dimRecord, elencoClienti, dimClienti);
	printf("Clienti Attivi:\n");
	stampaListaClienti(attivi);
	printf("\n\n");

	cattivi = illeciti(elenco, dimRecord, elencoClienti, dimClienti);
	printf("Illeciti:\n");
	stampaListaClienti(cattivi);


	free(elenco);
	freelist(attivi);
	freelist(cattivi);
	system("pause");
	return 0;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 6A di Mercoledì 14 Settembre 2011 – tempo a disposizione 2h

"info.txt":

```
abcde98765abcde 450 192.168.2.245
ghjkl12345ghjkl 460 100.0.0.0
abcde98765abcde 479 192.168.2.245
abcde98765abcde 480 200.222.2.2
abcde98765abcde 484 111.111.1.1
ghjkl12345ghjkl 514 100.0.0.0
ghjkl12345ghjkl 517 100.0.0.0
qwert54321qwert 531 221.113.113.112
qwert54321qwert 534 98.99.81.82
```

"archivio.txt":

```
abcde98765abcde Federico Chesani
ghjkl12345ghjkl Paola Mello
zxcvb56789zxcvb Stefano Bragaglia
qwert54321qwert Carlo Giannelli
```