Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

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: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire di tanto in tanto *"Rebuild All"*.

Una ditta di traslochi pianifica la propria attività memorizzando su un file di testo denominato "**`lavori.txt`**" i vari impegni lavorativi di trasloco. In particolare, su ogni riga del file, la ditta memorizza il nome del **cliente** (una stringa di al più 255 caratteri utili, senza spazi), uno spazio separatore e poi il **giorno** in cui deve effettuare il trasloco (un intero, compreso tra 1 e 365, rappresentante il giorno dall'inizio dell'anno), uno spazio separatore e infine la **stima** del numero di ore necessarie ad effettuare il trasloco (un intero, compreso tra 1 e 8).
I dipendenti della ditta possono lavorare al massimo otto ore nello stesso giorno. A volte però (a causa di un bug) il file dei lavori previsti contiene più traslochi per lo stesso giorno, e la somma eccede il limite delle otto ore giornaliere. Lo scopo del programma è identificare tali situazioni.

Esercizio 1 – Strutture dati Trasloco, e funzioni di lett./scritt. (mod. element.h/c e traslochi.h/.c)

Si definisca un'opportuna struttura dati **`Trasloco`**, al fine di rappresentare i dati relativi ad un trasloco, come da descrizione precedente.
Si definisca la funzione:

```
list leggiTraslochi(char* fileName);
```

che, ricevuto in ingresso il nome di un file contenente i traslochi previsti, legga tali informazioni e restituisca l'elenco dei traslochi tramite una lista di strutture dati di tipo **`Trasloco`**. In caso di errore nell'apertura del file la funzione deve stampare un messaggio di errore, e poi deve restituire una lista vuota.
Si definisca poi la procedura:

```
void stampaTraslochi(list elenco);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Trasloco`**, stampi a video tale elenco.
Si realizzi poi una funzione:

```
list ordina(list elenco);
```

che, , ricevuta in ingresso una lista di strutture dati di tipo **`Trasloco`**, restituisca una nuova lista contenente le stesse strutture dati fornite in ingresso, ma ordinate in base al giorno di trasloco, in ordine crescente. A parità di giorno, le strutture devono essere ordinate in ordine crescente secondo la durata del trasloco, e in caso di parità ulteriore, in ordine lessicografico in base al nome del cliente.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Esercizio 2 – Ricerca dei beni e ordinamento (modulo element.h e traslochi.h/.c)

Si definisca una funzione:

```
list filtra(list elenco, int giorno);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Trasloco`** e un giorno specifico, restituisca una nuova lista contenente le stesse strutture dati relative ai traslochi che saranno effettuati nel giorno specificato come parametro.
Si definisca una funzione:

```
int * giorni(list elenco, int * dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Trasloco`**, restituisca un vettore di interi allocato dinamicamente (eventualmente non della minima dimensione possibile), contenente i giorni in cui sono previsti dei traslochi, senza ripetizioni. Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione (logica) del vettore. Il vettore restituito deve essere ordinato in senso crescente (si usi un algoritmo di ordinamento a scelta tra quelli visti a lezione). Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

Esercizio 3 –Identificazione dei beni effettivamente acquistabili (modulo traslochi.h/traslochi.c)

Si definisca quindi una funzione:

```
list elimina(list elenco);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Trasloco`**, restituisca una nuova lista contenente le stesse strutture dati fornite in ingresso, dalla quale siano stati rimossi tutti quei traslochi che, sommata la loro durata, eccederebbero il limite giornaliero. In particolare, se per un giorno si prevede di eccedere il limite, si devono eliminare tutti i traslochi previsti per quel giorno.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento della funzione di cui sopra.

Esercizio 4 – Ricerca dei traslochi da effettuare, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che chieda all'utente di specificare un giorno specifico, e stampi a video i traslochi che si dovranno effettuare quel giorno specifico. Il programma in particolare dovrà leggere l'elenco dei traslochi da un file di nome "lavori.txt" (usando le funzioni di cui al punto 1), eliminare quei lavori che potrebbero eccedere il limite giornaliero di ore lavorative (usando la funzione di cui al punto 3), e stampare a video i lavori rimanenti previsti per il giorno specificato.

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, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

"element.h":

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

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_CLIENTE 256


typedef struct {
	char cliente[DIM_CLIENTE];
	int giorno;
	int stimaOre;
} Trasloco;


typedef Trasloco element;

int compareTrasloco(Trasloco t1, Trasloco t2);

#endif
```

"element.c":

```
#include "element.h"


int compareTrasloco(Trasloco t1, Trasloco t2) {
	int result = t1.giorno-t2.giorno;
	if (result == 0)
		result = t1.stimaOre - t2.stimaOre;
	if (result == 0)
		result = strcmp(t1.cliente, t2.cliente);
	return result;
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – 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 showlist(list l);
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);
}

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 (compareTrasloco(el, patt->value)<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, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

"traslochi.h":

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

#ifndef _TRASLOCHI
#define _TRASLOCHI

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

list leggiTraslochi(char* fileName);
void stampaTraslochi(list elenco);
list ordina(list elenco);

int * giorni(list elenco, int * dim);
list filtra(list elenco, int giorno);

list elimina(list elenco);
#endif
```

"traslochi.c":

```
#include "traslochi.h"



list leggiTraslochi(char* fileName) {
	FILE * fp;
	Trasloco temp;
	list result = emptylist();

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file %s.\n", fileName);
		return result;
	}

	while (fscanf(fp, "%s%d%d", temp.cliente, &(temp.giorno), &(temp.stimaOre)) == 3)
		result = cons(temp, result);
	fclose(fp);
	return result;
}

void stampaTraslochi(list elenco) {
	Trasloco temp;
	while(!empty(elenco)) {
		temp = head(elenco);
		printf("Cliente %s, previsto per il giorno %d, serviranno %d ore.\n",
temp.cliente, temp.giorno, temp.stimaOre);
		elenco = tail(elenco);
	}
	printf("\n\n");
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

```
list ordina(list elenco) {
	list result = emptylist();
	while (!empty(elenco)) {
		result = insord_p(head(elenco), result);
		elenco = tail(elenco);
	}
	return result;
}


list filtra(list elenco, int giorno) {
	list result = emptylist();
	while (!empty(elenco)) {
		if (head(elenco).giorno == giorno)
			result = cons(head(elenco), result);
		elenco = tail(elenco);
	}
	return result;
}


int trovaPosMax(int v[], int n){
	int i, posMax=0;
	for (i=1; i<n; i++)
		if (v[posMax]<v[i]) posMax=i;
	return posMax;
}
void scambia(int*a, int*b) {
	int temp;
	temp=*a;*a=*b;*b=temp;
}
void naiveSort(int v[], int n){
	int p;
	while (n>1) {
	p = trovaPosMax(v,n);
	if (p<n-1) scambia(&v[p],&v[n-1]);
	n--;
	}
}


int * giorni(list elenco, int * dim) {
	int count;
	int * result;
	list temp1;
	list temp2;

	*dim = 0;
	count = 0;
	temp1 = elenco;
	while (!empty(temp1)) {
		count++;
		temp1 = tail(temp1);
	}

	result = (int*) malloc(sizeof(int) * count);
```

```
        temp1 = ordina(elenco);
        temp2 = temp1;
        if (!empty(temp1)) {
                result[*dim] = head(temp1).giorno;
                (*dim)++;
                temp1 = tail(temp1);
                while (!empty(temp1)) {
                        if (head(temp1).giorno != result[*dim-1]) {
                                result[*dim] = head(temp1).giorno;
                                (*dim)++;
                        }
                        temp1 = tail(temp1);
                }
        }
        freelist(temp2);
        naiveSort(result, *dim);
        return result;
}



list elimina(list elenco) {
        list temp1;
        list temp2;
        list result = emptylist();
        int durata;
        int * gg;
        int dim, i;

        gg = giorni(elenco, &dim);
        for (i=0; i<dim; i++) {
                temp1 = filtra(elenco, gg[i]);
                temp2 = temp1;
                durata = 0;
                while (!empty(temp1)) {
                        durata = durata + head(temp1).stimaOre;
                        temp1 = tail(temp1);
                }
                if (durata<=8) {
                        temp1 = temp2;
                        while (!empty(temp1)) {
                                result = cons(head(temp1), result);
                                temp1 = tail(temp1);
                        }
                }
                freelist(temp2);
        }

        free(gg);
        return result;
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

"main.c":

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

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


int main() {

	{ // test es. 1
		list elenco;
		list elencoOrd;
		elenco = leggiTraslochi("lavori.txt");
		stampaTraslochi(elenco);
		elencoOrd = ordina(elenco);
		stampaTraslochi(elencoOrd);
		freelist(elenco);
		freelist(elencoOrd);
	}
	{ // test es. 2
		list elenco;
		list test;
		int * ggg;
		int dim, i;
		elenco = leggiTraslochi("lavori.txt");
		test = filtra(elenco, 135);
		stampaTraslochi(test);
		ggg = giorni(elenco, &dim);
		for (i=0; i<dim; i++)
			printf("%d, ", ggg[i]);
		freelist(elenco);
		freelist(test);
	}
	{ // test es. 3
		list elenco;
		list test;
		elenco = leggiTraslochi("lavori.txt");
		test = elimina(elenco);
		stampaTraslochi(test);
		freelist(elenco);
		freelist(test);
	}
	{ // test es. 4
		list elenco;
		list test;
		list temp;
		int unGiorno;
		elenco = leggiTraslochi("lavori.txt");
		test = elimina(elenco);
		temp = test;
		printf("A quale giorno si e' interessati? ");
		scanf("%d", &unGiorno);
		while (!empty(temp)) {
			if (head(temp).giorno == unGiorno)
```

```
                        printf("Cliente %s, previsto per il giorno %d, serviranno %d
ore.\n", head(temp).cliente, head(temp).giorno, head(temp).stimaOre);
                    temp = tail(temp);
            }
            freelist(elenco);
            freelist(test);
      }
      system("pause");
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 4A di Giovedì 12 Giugno 2014 – tempo a disposizione 2h

"lavori.txt":

```
Chesani 68 5
Giannelli 47 6
Mello 47 4
Chesani 135 7
Mello 135 1
```