Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – 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"*.

Un negozio di accessori auto è specializzato in catene da neve, che vende su misura per ogni tipo di pneumatico: in particolare, le catene sono classificate in base al diametro della ruota su cui si possono montare, e in base alla larghezza del battistrada della ruota. Ad esempio, le catene indicate come "15/205" si montano su ruote che abbiano un diametro di almeno 15 pollici, e larghezza di almeno 205 mm.
Il negozio tiene traccia delle confezioni che ha in vendita tramite un file di testo, dove per ogni confezione vengono annotati su una singola riga i dati della confezione. In particolare, nell'ordine specificato, compaiono: il **diametro** della ruota su cui si possono montare le catene (un intero); separato da uno spazio, la **larghezza** del battistrada (un intero); separato da uno spazio, il **prezzo** della confezione (un float); ed infine, ancora separato da uno spazio, il nome dell'**azienda** che ha prodotto la confezione (una stringa di al più 255 caratteri utili, contenente spazi). Tutte le righe tranne l'ultima sono terminate da un carattere newline '\n'.
Si noti infine che ogni riga è relativa ad una singola confezione di catene da neve: due confezioni identiche compariranno quindi su due righe distinte, con tutti i valori duplicati e identici.
Si veda, a titolo di esempio, il file di testo "catene.txt" fornito nello StartKit.

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

Si definisca una opportuna struttura dati **`Catena`** per memorizzare una singola confezione di catene da neve: si memorizzi quindi il diametro e la larghezza della ruota adatte, il prezzo e il nome della ditta produttrice.

Si definisca la funzione:

```
Catena leggiUno(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo **`FILE`**, legga i dati contenuti su una sola riga e li restituisca tramite una struttura dati di tipo **`Catena`**. Qualora vi siano problemi nella lettura, la funzione deve restituire una struttura dati con tutti i campi numerici pari a zero, e il campo stringa inizializzato alla stringa vuota.

Si definisca la funzione:

```
Catena * leggiTutte(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga da tale file i dati di tutte le confezioni di catene da neve e li restituisca tramite un array di strutture dati di tipo **`Catena`** allocato dinamicamente (della dimensione minima necessaria). Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a NULL, e zero come valore della dimensione.

Si definisca la procedura:

```
void stampa(Catena * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Catena`**, e la sua dimensione **`dim`**, stampi a video i dati delle confezioni di catene disponibili in negozio.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

Esercizio 2 –Ordinamento e conta delle confezioni (moduli element.h/.c e catene.h/.c)

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Catena`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine crescente in base al parametro "diametro"; a parità di diametro, in ordine crescente in base al parametro "larghezza"; a parità di larghezza, in ordine crescente in base al prezzo; a parità di prezzo, in ordine lessicografico in base al nome del produttore. A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "naive sort" visto a lezione.

Si definisca una struttura dati di nome **`Prodotto`**, che contenga due campi: una struttura dati di tipo **`Catena`**, ed un intero. Il campo di tipo intero servirà per memorizzare quante confezioni di un certo tipo di catene da neve sono disponibili in negozio. Ad esempio, il negozio potrà avere 5 confezioni di catene 15/205, e 3 confezioni di catene 18/225.

Si definisca poi una funzione:

```
list conta(Catena * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Catena`**, e la sua dimensione **`dim`**, restituisca in uscita una lista di strutture dati di tipo **`Prodotto`**, contenente l'elenco delle tipologie di confezioni di catene, e il numero di confezioni disponibili in negozio.

Esercizio 3 – Filtraggio modelli (modulo catene.h/catene.c)

I clienti si recano in negozio per comprare le catene: specificano diametro e larghezza delle ruote, e il commesso deve proporre loro i modelli di catene da neve adeguati. A tal scopo, si sviluppi una funzione:

```
list filtra(int diametro, int larg, list disponibili);
```

che, ricevuti in ingresso diametro e larghezza richiesti dal cliente, e la lista delle confezioni disponibili in negozio, restituisca una nuova lista con l'elenco delle strutture dati di tipo **`Prodotto`** che riguardano confezioni di catene **compatibili** con le richieste fornite dal cliente. Si noti che catene da neve "piccole" non possono essere montate su ruote "grandi"; al contrario, una catena "grande" può essere montata su una ruota piccola, a patto che la differenza di dimensioni sia al massimo del 10%. Si definiscono quindi catene "**compatibili**" le catene che sono previste per ruote con diametro al massimo 10% più grande della ruota specificata, e larghezza al massimo 10% maggiore della larghezza specificata.
La lista restituita in uscita dovrà essere ordinata in senso decrescente in base al prezzo: in testa alla lista dovranno essere specificate le catene più costose.

Esercizio 4 Stampa dei risultati, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dal file tutte le confezioni di catene da neve disponibili in negozio, chieda all'utente di specificare un diametro ed una larghezza, e stampi a video la lista delle catene da neve compatibili, proponendo al cliente per prime le catene da neve più costose.

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, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_AZIENDA 256

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

typedef struct {
	int diametro;
	int larghezza;
	float prezzo;
	char azienda[DIM_AZIENDA];
} Catena;


typedef struct {
	Catena modello;
	int quant;
} Prodotto;

typedef Prodotto element;
int compare(Catena c1, Catena c2);
int compatibili(int diametro, int larghezza, Catena C1);
#endif
```

"element.c":

```
#include "element.h"
int compare(Catena c1, Catena c2) {
	int result;
	result = c1.diametro - c2.diametro;
	if (result == 0) {
		result = c1.larghezza - c2.larghezza;
		if (result == 0) {
			if (c1.prezzo<c2.prezzo) result = -1;
			else
				if (c1.prezzo > c2.prezzo) result = 1;
				else result = 0;
			if (result == 0)
				result = strcmp(c1.azienda, c2.azienda);
		}
	}
	return result;
}

int compatibili(int diametro, int larghezza, Catena c1) {
	if (diametro>=c1.diametro
		&& diametro <= c1.diametro*1.1
		&& larghezza>=c1.larghezza
		&& larghezza <= c1.larghezza*1.1)
		return 1;
	else
		return 0;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – 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);
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

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

void showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%d: %d %d %f %s\n",
                     temp.quant, temp.modello.diametro, temp.modello.larghezza,
temp.modello.prezzo, temp.modello.azienda);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}

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 (el.modello.prezzo > patt->value.modello.prezzo)
                     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, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

"catene.h":

```
#ifndef _CATENE_H
#define _CATENE_H

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

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

// Es. 1
Catena leggiUno(FILE * fp);
Catena * leggiTutte(char * fileName, int * dim);
void stampa(Catena * v, int dim);

// Es. 2
void ordina(Catena * v, int dim);
list conta(Catena * v, int dim);

// Es. 3
list filtra(int diametro, int larg, list disponibili);

#endif
```

"catene.c":

```
#include "catene.h"
// Es. 1
Catena leggiUno(FILE * fp) {
	Catena result;
	char ch;
	int i=0;
	if (fscanf(fp, "%d%d%f", &(result.diametro), &(result.larghezza), &(result.prezzo))
== 3) {
		ch = fgetc(fp); // elimino lo spazio di separazione
		do {
			ch=fgetc(fp);
			if (ch!='\n' && ch!= EOF && i<DIM_AZIENDA-1) {
				result.azienda[i] = ch;
				i++;
			}
		} while ( ch != '\n' && ch!=EOF && i<DIM_AZIENDA-1);
		result.azienda[i] = '\0';
	}
	else {
		result.diametro = 0;
		result.larghezza = 0;
		result.prezzo = 0;
		strcpy(result.azienda, "");
	}
	return result;
}

Catena * leggiTutte(char * fileName, int * dim) {
	FILE * fp;
```

```
	Catena * result = NULL;
	Catena temp;
	int count = 0;

	*dim = 0;
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		temp = leggiUno(fp);
		while (temp.diametro!=0 || temp.larghezza!=0 || temp.prezzo!=0) {
			count++;
			temp = leggiUno(fp);
		}
		result = (Catena*) malloc(sizeof(Catena)*count);
		rewind(fp);
		temp = leggiUno(fp);
		while (temp.diametro!=0 || temp.larghezza!=0 || temp.prezzo!=0) {
			result[*dim] = temp;
			*dim = *dim + 1;
			temp = leggiUno(fp);
		}
		fclose(fp);
	}
	else {
		printf("Errore nell'apertura del file: %s\n", fileName);
	}
	return result;
}

void stampa(Catena * v, int dim) {
	int i=0;

	for(i=0; i<dim; i++) {
		printf("%d %d %f %s\n",
			v[i].diametro, v[i].larghezza, v[i].prezzo, v[i].azienda);
	}
	return;
}

// Es. 2
void scambia(Catena *a, Catena *b) {
	Catena tmp = *a;
	*a = *b;
	*b = tmp;
}
int trovaPosMax(Catena v[], int n){
	int i, posMax=0;
	for (i=1; i<n; i++)
		if (compare(v[posMax],v[i])<0)
			posMax=i;
	return posMax;
}
void ordina(Catena * v, int n) {
	int p;
	while (n>1) {
		p = trovaPosMax(v,n);
		if (p<n-1)
			scambia(&v[p],&v[n-1]);
```

```
            n--;
      }
}

list conta(Catena * v, int dim) {
      list result = emptylist();
      int i = 0;
      Prodotto temp;

      ordina(v, dim);
      if (dim>0) {
            temp.modello = v[i];
            temp.quant = 1;
            for (i=1; i<dim; i++) {
                  if (compare(v[i], temp.modello) == 0)
                        temp.quant = temp.quant + 1;
                  else {
                        result = cons(temp, result);
                        temp.modello = v[i];
                        temp.quant = 1;
                  }
            }
            result = cons(temp, result); //aggiungo l'ultimo modello nella lista
      }

      return result;
}


// Es. 3
list filtra(int diametro, int larg, list disponibili) {
      list result;
      Prodotto temp;

      result = emptylist();
      while (!empty(disponibili)) {
            temp = head(disponibili);
            if (compatibili(diametro, larg, temp.modello))
                  result = insord_p(temp, result);
            disponibili = tail(disponibili);
      }
      return result;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

"main.c":

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

#include "element.h"
#include "catene.h"

int main() {
	// Es. 1
	{
		Catena * v;
		int dim;
		v = leggiTutte("catene.txt", &dim);
		stampa(v, dim);
		printf("\n\n\n");
		free(v);
	}
	// Es. 2
	{
		Catena * v;
		int dim;
		list elenco;
		v = leggiTutte("catene.txt", &dim);
		ordina(v, dim);
		stampa(v, dim);
		printf("\n\n\n");
		elenco = conta(v, dim);
		showlist(elenco);
		printf("\n\n\n");
		free(v);
		freelist(elenco);
	}
	// Es. 3
	{
		Catena * v;
		int dim;
		list elenco;
		list prod;
		int diametro;
		int larghezza;

		v = leggiTutte("catene.txt", &dim);
		elenco = conta(v, dim);
		printf("Inserire diametro e larghezza della gomma: ");
		scanf("%d%d", &diametro, &larghezza);

		prod = filtra(diametro, larghezza, elenco);
		showlist(prod);
		free(v);
		freelist(elenco);
		freelist(prod);
	}
	return 0;
}
```

Fondamenti di Informatica T-1, 2017/2018 – Modulo 2

Prova d'Esame 1A di Giovedì 11 Gennaio 2018 – tempo a disposizione 2h

"catene.txt":

```
65 155 12.90 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
65 175 13.50 Neve Go Go Unlimited Inc.
65 175 19.90 Catene Sport s.r.l.
80 175 15.90 Neve Go Go Unlimited Inc.
60 185 14.90 Neve Go Go Unlimited Inc.
60 185 19.90 Catene Sport s.r.l.
55 195 22.90 Catene Sport s.r.l.
55 195 21.90 Neve Go Go Unlimited Inc.
65 165 12.89 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
80 175 15.90 Neve Go Go Unlimited Inc.
80 175 15.90 Neve Go Go Unlimited Inc.
80 175 15.90 Neve Go Go Unlimited Inc.
55 195 22.90 Catene Sport s.r.l.
55 195 22.90 Catene Sport s.r.l.
55 195 22.90 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
65 165 12.89 Catene Sport s.r.l.
```