Laboratorio Informatica A

Problema 1

Dichiarare e popolare una lista composta da tre interi inseriti dall'utente. Stampare a video la somma di tutti gli elementi della lista così costruita. Evitare di usare le funzioni viste in aula.

Problema 2

Generalizzare il problema precedente per permettere l'inserimento di un numero *n* di elementi definito dall'utente a runtime. Si utilizzino le funzinioni viste a lezione per operare sulle liste-

Problema 3

Scrivere una funzione f (e main per chiamarla) che riceve un array di 5 interi e alloca una lista di 5 elementi contenente gli elementi dell'array in ordine inverso. Si utilizzino pure le funzioni viste a lezione.

Problema 4

La FPF Inc. (Fictional Pigs Farm) si dedica da sempre all'allevamento di maialini da palcoscenico più o meno famosi, e conserva i dati dei suoi piccoli divi in un vettore (di cui solo una parte è utilizzata, pari a num_maialini: gli elementi di indice da num_maialini a N-1 non sono usati):

typedef struct { int giorno; int mese; int anno; } Data;

typedef struct { char nome[20]; Data datanascita; float peso; int popolarita; } Maialino;

typedef struct { int num_maialini; Maialino pigs[N]; } Allevamento;

I maialini non sono elencati nell'Allevamento in un alcun ordine particolare, ma la ditta necessita di scandirli in ordine di ognuna delle loro caratteristiche (cioè in ordine di nome, di data di nascita, di popolarità, e di peso corporeo), a seconda delle diverse necessità applicative (rispettivamente: appello nominale, trattamento pensionistico, merchandising, utilizzo culinario qualora la popolarità cali eccessivamente). A tal fine, si utilizzano quattro distinti vettori di puntatori, in cui ogni puntatore punta a un maialino specifico. In questo modo si rappresentano quattro diversi ordinamenti indipendenti degli elementi di uno stesso insieme (di maiali) senza dover replicare tutti i dati ad essi relativi, ma replicando solo i puntatori:

Maialino * ord_alfabetico[N], ord_data[N], ord_pop[N], ord_peso[N];

Si codifichi un programma che costruisca correttamente i quattro vettori di puntatori. Il programma deve dapprima stampare tutti i dati dei maialini, nell'ordine in cui si trovano nell'allevamento, poi ordinare i puntatori nei vari vettori, e visualizzare i maialini secondo i diversi ordinamenti. Si utilizzi (anche) la funzione int confronta(Data d1, Data d2) che restituisce 0 se le due date sono uguali, 1 se in ordine cronologico crescente, -1 se in ordine inverso.

Di seguito delle strutture dati già inizializzate:

```
#include <stdio.h>
#define N 100 // fino a 100 maialini in un allevamento
typedef struct {
        int giorno, mese, anno;
} Data;
typedef struct {
        char nome[20];
        Data datanascita;
                float peso;
        int popolarita;
} Maialino;
typedef struct {
        int numero_maialini;
        Maialino pigs[N];
} Allevamento;
int confronta( Data, Data );
int main () {
         Allevamento a = { 7,  "Porky Pig", {12,7,1936}, 33.50, 85 ,
                                                "Miss Piggy", {17,12,1974}, 23.95, 170 ,
                                                "Babe", {23,1,1996}, 18.80, 250 ,
                                        "Pumbaa", {14,11,1994}, 79.99, 1690 ,
                                        "Peppa", {31,5,2004}, 12.15, 8500 ,
                                        "Hamm", {12,7,1968}, 19.05, 290 ,
                                        "Piglet", {22,4,1926}, 9.30, 1260  };
        Maialino  * ord_alfabetico[N] = { NULL },   // inizialmente, per "sicurezza", tutti a NULL
        * ord_data[N] = { NULL },          * ord_peso[N] = { NULL },          * ord_pop[N] = { NULL };
         /* codice! */
         return 0;
}
```

Problema 5

Costruite un programma per gestire una lista di film. Per immagazzinare i film dovete usare la seguente struttura:

typedef struct _movie { char title[200]; char type[200]; int year; } movie;

Per immagazzinare i film che vengono inseriti dall'utente dovete usare un array. In questo caso, il dato contenuto in ogni casella dell'array non sarà un semplice char o un int, ma una struttura di tipo movie. Nell'array i nuovi film devono essere inseriti in modo ordinato a seconda dell'anno in cui è stato girato il film (prima i film più vecchi poi quelli più nuovi).

movie film[100];

int numFilm;

numFilm servirà a sapere quante caselle dell'array sono effettivamente usate

Per questo progetto implementate le funzioni che devono essere richiamate opportunamente in un apposito main di test:

• int add(movie f[],int numFilm) //chiede i dati per un nuovo film, lo aggiunge nella posizione corretta e restituisce il numero di film aggiornato.

• void print(movie f[],int numFilm) //stampa la lista dei film.

• movie search(movie f[],int numFilm, char * title) //cerca un film nella lista in base al titolo e restituisce l'elemento dell'array che contiene tale film.

• int remove(movie f[],int numFilm,char * title) //rimuove un film senza lasciare buchi e restituisce il numero di film aggiornato.

Problema 6

Costruite un programma per gestire una lista di film. Per immagazzinare i film dovete usare la seguente struttura

Sia data la seguente definizione di lista

```
typedef struct nodo_t {
        int e;
        nodo_t *next;
        nodo_t  *down;
} nodo;
```

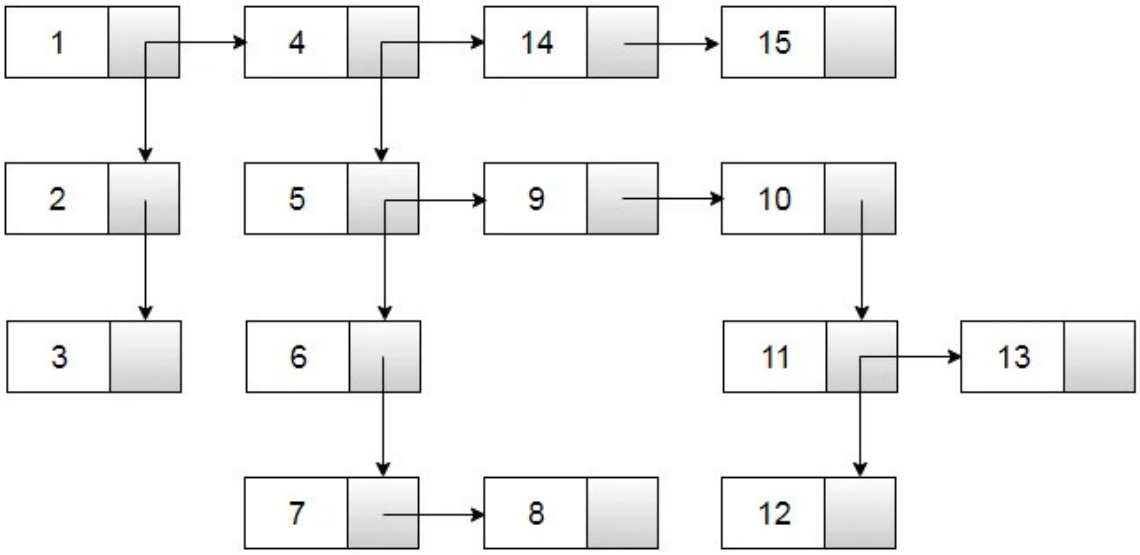
dove attraverso il puntatore down si contempla la possibilità di annidare liste una dentro l'altra a qualsiasi livello di profondità. Si richiede di scrivere una funzione flatten che data in ingresso una lista della tipologia sopra specificata ne restituisce la sua versione "appiattita".

Per la versione "appiattita" si utilizzi la seguente struttura dati:

```
typedef struct nodo_f_t{
        int e;
```

```
        nodof_t *next;
}nodo_f;
```

Esempio:



1->2->3->4->5->6->7->8->9->10->11->12->13->14->15