Laboratorio Informatica A

Problema 1

Scrivere una funzione che riceve una matrice M di interi e un intero K e restituisce 1 se la matrice ha una riga i cui valori sono tutti minori di K. Si usi il file lab6_1.c allegato per scrivere e testare la funzione.

Problema 2

Data la struct che definisce un numero complesso:

typedef struct Complex{ double Re, Im; } Complex;

Si definisca una funzione GetComplesso(Complex * n) che legga un numero complesso passato per riferimento
Si definisca una funzionericorsiva che riceve un array di complessi e restituisce il numero complesso di modulo massimo.
Complex MaxComplessoModulo(Complex * array)

Problema 3

Scrivere una funzione analizzaMatrice che, data una matrice di interi A di dimensioni N x N (con N costante definita con #define N ...) stampa a schermo tutti gli elementi che risultano contemporaneamente massimo della riga e della colonna e di entrambe le diagonali di appartenenza.

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.
Nel file maialini.c trovate delle strutture dati già inizializzate.

Problema 5

Si vuole realizzare un programma che permetta di giocare al gioco del 15. In tale gioco, una scacchiera 4x4 contiene 15 pezzi (numerati da 1 a 15) ed una casella vuota (rappresentata da uno 0). Il giocatore ad ogni mossa può spostare uno dei pezzi adiacenti alla casella vuota nella casella vuota stessa. Le mosse sono specificate indicando il numero del pezzo da spostare e il gioco continua fino a quando tutti i numeri non compaiono nell'ordine numerico corretto. Nell'esempio a lato le mosse possibili sono: 2, 1, 7, 3. Se il giocatore sceglie la mossa 3, le mosse possibili diventano: 3, 10, 15, 14. Nella risoluzione dell'esercizio si implementino le seguenti funzioni:

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

- int valida (int gioco[N][N], int mossa) che riceve la scacchiera e una mossa e restituisce 1 se la mossa è valida, 0 altrimenti.
- void muovi (int gioco[N][N], int mossa) che riceve la scacchiera e una mossa e aggiorna la scacchiera in base alla mossa effettuata.
- int risolto (int gioco[N][N]) che riceve la scacchiera e restituisce 1 se il gioco è stato risolto, 0 altrimenti.
- void stampa (int gioco[N][N]) che riceve la scacchiera e la stampa a video.

Problema 6

Si consideri un videogame dal nome “Godzilla Frenzy”. In tale gioco, il mondo è una griglia MxN e in ogni cella l'utente può costruire delle case o delle strade. Un esempio di mondo è la griglia a lato.

X	0	1	2	3	4	5	6	7	8	9
0	C	C	C	C	C	C	C	C	C	C
1	C	C	C	C	C	C	C	S	C	C
2	S	S	S	S	S	S	S	S	C	C
3	C	S	C	C	C	C	C	S	C	C
4	C	C	S	C	C	C	C	S	C	C
5	C	C	S	S	S	C	C	S	S	S
6	S	S	S	S	S	C	C	S	C	C
7	C	C	S	C	C	S	S	S	C	C
8	C	C	S	C	C	C	C	S	C	C
9	C	C	S	C	C	C	C	S	C	C

Mentre l'utente costruisce la sua città, Godzilla può arrivare improvvisamente e distruggere ciò che è stato costruito. Tuttavia, Godzilla si può muovere solo in orizzontale e in verticale e ha una paura tremenda delle strade e non vuole assolutamente attraversarle. Pertanto, Godzilla atterra su una casella scelta a caso e si può muovere solo sulle case, quando trova una strada (o il bordo della griglia) è costretto ad indietreggiare. Implementare una funzione ***ricorsiva*** che permette di calcolare, a partire da una casella di partenza (quella in cui atterra Godzilla), su quali caselle il mostro si può muovere. Ad esempio, se consideriamo la griglia precedente e la casella di coordinate (4,4), la funzione dovrà marcare nella griglia le seguenti caselle

X	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	C	C	C	C	C	C	C	C	C	C
1	C	C	C	C	C	C	C	**S**	C	C
2	**S**	**S**	**S**	**S**	**S**	**S**	**S**	**S**	C	C
3	C	**S**	*g*	*g*	*g*	*g*	*g*	**S**	C	C
4	C	C	***S***	*g*	*g*	*g*	*g*	**S**	C	C
5	C	C	**S**	**S**	**S**	*g*	*g*	**S**	S	S
6	**S**	**S**	**S**	**S**	**S**	*g*	*g*	**S**	C	C
7	C	C	**S**	C	C	**S**	**S**	**S**	C	C
8	C	C	**S**	C	C	C	C	**S**	C	C
9	C	C	**S**	C	C	C	C	**S**	C	C