Laboratorio Informatica A - Incontro 5

Problema 1

Scrivere una funzione ***ricorsiva*** (e un main che la testa) che chieda un intero positivo n all'utente e in risposta stampi i primi n numeri della serie tribonacci. La successione tribonacci è una variante della successione di Fibonacci. Mentre quest'ultima viene definita fissando i primi due termini e chiedendo che ogni termine sia la somma dei due che la precedono, la successione tribonacci è definita come la sequenza illimitata di termini t(n) per i quali si assume ...

t(-2) = t(-1) = t(0) = 0 t(1) = 1

e che per ciascuno dei successivi termini si chiede che sia uguale alla somma dei tre termini precedenti.

I valori dei suoi primi termini, a partire da quello di indice 1, sono 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504, 927, 1705, 3136, 5768

Problema 2

Si consideri una matrice (o griglia) di caratteri di dimensione NxN, definita come segue:

typedef char griglia[N][N];

(a) Si codifichi in C una funzione char ennuplaVert(griglia g) che, se esiste una colonna in g costituita da un allineamento verticale di N caratteri tutti uguali e diversi da '\0', restituisce il carattere presente su tale colonna, altrimenti restituisce il carattere '\0'. Si trascuri il caso in cui più colonne godano della proprietà da verificare, considerando il primo allineamento rilevato (ad es., il più a sinistra).

(b) Si indichi come ottenere una funzione char ennuplaOriz(griglia g) che effettui analoga verifica per un allineamento orizzontale, attraverso la minima modifica al codice della funzione precedente.

(c) Si codifichi in C una funzione char ennuplaDiag(griglia g) che verifica in modo analogo alle precedenti se vi è un allineamento su una delle due diagonali principali.

(d) Nel popolarissimo gioco del tic tac toe (tris in italiano) i giocatori tentano di disporre 3 simboli uguali su una griglia 3x3 apponendovi a turno delle 'X' (il simbolo del 1° giocatore) e delle 'O' (simbolo del 2°). Nella griglia a lato la partita è terminata in parità (come sempre succede, se nessun giocatore commette errori). Vogliamo generalizzare il gioco al caso NxN, usando le definizioni dei punti precedenti. Si codifichi in C una funzione int esito(griglia g) che restituisce 0 se la partita rappresentata da g non è finita, 1 se ha vinto il "giocatore X", 2 se ha vinto il "giocatore O", 3 se è un pareggio.

(e) Si consideri la seguente definizione di una partita rappresentata come sequanza di mosse, in cui ogni mossa è una coppia di interi compresi tra 0 e N-1 che indicano la riga e la colonna in cui i giocatori appongono i simboli. La prima mossa è del giocatore X, la seconda del giocatore O, … . Si assuma per semplicità che le mosse di una partita siano sempre valide (coordinate corrette, simboli non sovrapposti, …).
typedef struct mo { int r,c; } Mossa;
typedef Mossa Partita[1000];
Nell'array partita l'ultima mossa è seguita da una (finta) mossa di chiusura con r e c entrambi uguali a N

Si codifichi in C una funzione int gioca(griglia g, Partita p) che riceve come parametro una Partita e ne indica l'esito finale secondo le convenzioni del punto precedente (simulandone opportunamente l'esecuzione, applicando le mosse ad una griglia inizialmente vuota).

Problema 3

Il prontuario farmaceutico nazionale (PFN) elenca i farmaci mutuabili fornendone una breve descrizione. Nella definizione sottostante, il PFN può contenere fino a 40mila diverse commercializzazioni dei principi attivi disponibili, di ognuna indicando se è mutuabile o no (mutuabileSiNo). In un PFN non pieno, le posizioni non utilizzate sono "in fondo" e contraddistinte dal valore convenzionale 0 per barcode.

```
typedef struct f { unsigned long int barcode;
                   float dosaggio;
                   char nome[20], principioattivo[20], produttore[20]; } Farmaco;
typedef struct ep { Farmaco farmaco;
                    int mutuabileSiNo;
                    float prezzo_ufficiale;
                    char descrizione_e_indicazioni[250]; } ElementoProntuario;
typedef ElementoProntuario PFN[40000];
```

Si codifichi una funzione C … trova(…) che riceve in input un codice a barre (barcode) e un vettore di tipo PFN e restituisce l'indice dell'elemento del prontuario relativo al farmaco con quel codice, oppure il valore -1 se il codice non è presente nel prontuario

Si codifichi una funzione C … esente(…) che riceve come parametri un codice a barre (barcode) e un vettore di tipo PFN e, usando la funzione trova() per trovare il farmaco corrispondente, restituisce 1 se il farmaco identificato dal codice è mutuabile, 0 altrimenti.

Nelle farmacie più moderne per evitare di dare, per imperizia o distrazione, farmaci sbagliati ai clienti (e quindi per limitare lo stress di farmacisti e stagisti praticanti), oltre che per conservare meglio i farmaci, si è completamente robotizzato l'inserimento e il prelievo delle confezioni dagli scaffali, basando il processo sulla lettura ottica del codice a barre da parte di una periferica di input e sulla movimentazione meccanico-pneumatica delle scatole da parte di una periferica-attuatore di output, un robot che chiameremo Drugbot. Modelliamo per semplicità lo scaffale come un vettore tridimensionale di slot, ognuno dei quali può contenere una confezione di un qualsiasi farmaco (le dimensioni non contano) e assume valore pari al barcode del farmaco contenuto, oppure 0 se lo slot è vuoto. Modelliamo poi il registro

elettronico dei farmaci presenti nello scaffale come un array in cui ogni nodo indica la quantità di scatole disponibili di un dato farmaco e contiene a sua volta un array di posizioni, che sono tutte le coordinate degli slot dello scaffale che contengono confezioni di quel farmaco.

```
typedef unsigned long int Scaffale[HEIGHT][WIDTH][DEPTH]; //misure (costanti) dello scaffale
typedef struct p { int x,y,z; } Pos;
typedef struct ps { Pos pos;
                    int valido; } Posizione;
typedef Posizione Posizioni[1000];
typedef struct itm { unsigned long int barcode;
                     int qtà_disponibile;
                     Posizioni pos;
                     int valido; } Item;
typedef Item Registro[1000];
```

In entrambi gli array le caselle contengono un campo "valido" che è uguale a 1 se il dato contenuto contiene dati validi, uguale a 0 se è da considerarsi vuoto (cioè contenente dati casuali e non significativi).

Si codifichi la funzione int carica(unsigned long barcode, Scaffale s, Registro r) che aggiunge una confezione del farmaco specificato allo scaffale nel primo slot libero trovato e aggiorna il registro, inserendo la posizione dello slot occupato all'array delle posizioni e incrementando la quantità disponibile. Se l'operazione riesce, la funzione restituisce 1, se invece lo scaffale è pieno, non aggiorna i dati e restituisce 0. Si tratti, idealmente e se possibile, anche il caso in cui il farmaco non è ancora presente nel registro.

Problema 4

A Bronte siamo in piena stagione di raccolta dei pistacchi. Il raccolto dei vari appezzamenti di terreno dei soci della CAmPi (Cooperativa Amici del Pistacchio) è tutto smallato dall'unica smallatrice a disposizione (la smallatrice è la macchina che estrae i pistacchi dal loro mallo). Al mattino, dai vari appezzamenti (che per semplicità supporremo quadrati, disposti su una griglia quadrata NxN e identificati dalle coordinate x,y) i proprietari portano ciascuno una cesta di pistacchi da smallare. Le ceste sono accodate fino a saturare la capacità giornaliera della macchina (~50Kg/h, 600Kg al dì), dopodiché non sono più accettate, e saranno eventualmente riportate il giorno successivo. La smallatrice, di ultimissima generazione, al termine di ogni lavorazione indica quanti pistacchi erano contenuti nella cesta, sicché conoscendo il peso del contenuto della cesta si può stimare il calibro medio di ogni appezzamento. La coda delle ceste è modellata come segue:

```
typedef struct { int x, y;} Posizione;
typedef struct c { Posizione appezzamento; float peso; int valido } Cesta;  //valido è =1 se la struct contiene dati, è =0 se è da considerarsi vuota
typedef Cesta Coda[N];
```

Si codifichi in C la funzione di prototipo Coda accoda(Posizione p, float peso, Coda c), che se la capacità della smallatrice non è ancora saturata aggiunge alla coda la cesta descritta dai parametri.

Col procedere del raccolto, si tiene traccia del calibro medio dei pistacchi di ogni appezzamento su una mappa, modellata come matrice di float (ogni cella contiene inizialmente -1.0 e poi il calibro medio dell'appezzamento di coordinate corrispondenti).

```
typedef float Mappa[N][N];
```

Si codifichi in C la funzione di prototipo Coda smalla(Coda c, Mappa m) che

- preleva una cesta dalla coda,
- ne dispone la lavorazione invocando la funzione **int** start(Cesta cs, int numCeste) che restituisce il numero di pistacchi smallati,
- calcola il calibro medio,
- aggiorna la mappa, e infine
- restituisce la coda modificata.

La funzione start() è da considerarsi già implementata. Si badi a gestire il prelievo in modo che la coda si comporti effettivamente come una coda.

La cooperativa vuole sempre sapere se gli appezzamenti che hanno prodotto i pistacchi migliori (cioè di calibro maggiore) sono concentrati in una sola zona oppure sono sparsi su tutto il territorio. In particolare, si vuole valutare se i *k* appezzamenti col calibro maggiore sono contenuti in un'area quadrata di lato k, oppure sono più sparpagliati. *Ad esempio, nelle mappe a lato si vede che per k=3 nella mappa in alto esiste un "quadrato" di lato k che contiene i k valori più elevati, mentre un tale quadrato non esiste in quella in basso.*

Si codifichi in C la funzione **int** concentrati(Mappa m, **int** k) che restituisce **1** se i k valori più elevati in m stanno in un quadrato di lato k, 0 altrimenti. *Si suggerisce di definire e creare una semplice struttura dati di supporto per tener traccia, durante la scansione della matrice, dei valori più elevati e delle loro coordinate, per poi ragionare delle dimensioni del rettangolo di inviluppo basandosi sui dati di supporto (valutando le coordinate degli appezzamenti selezionati)*.