Settimana 3 Esercizi per casa

Portare lo sviluppo all'incontro del proprio gruppo, giovedi 10/venerdi 11 febbraio.
Prescrizione minima-minima: es.1, es 2 punto 1
(e' fortemente raccomandato di provare anche i punti 2 e 3.).

ES 1 La lepre e la tartaruga *(mini-esercizio sul passaggio dei parametri)*
Scrivere un programma che simuli la corsa della lepre e della tartaruga.
La lepre raggiunge il traguardo in 10 passi (balzi), la tartaruga in 12.
Vince il primo dei due che taglia il traguardo .

La distanza dal traguardo di ciascun contendente e' contenuta nelle variabili
int lepre, int tartaruga.

I contendenti devono essere fatti avanzare chiamando *la funzione muovi*, che prende come parametri la posizione della lepre e la posizione della tartaruga.

La funzione muovi usa rand()%2.
Se esce 1, la lepre avanza di un passo.
Se esce 0, la tartaruga avanza di un passo.

Il programma principale chiama muovi(lepre, tarta)
fino a quando uno dei due supera il traguardo.
Ad ogni chiamata di muovi(lepre,tarta), il programma stampa
la distanza dal traguardo (e' una cronaca minuto per minuto…) ed
alla fine, il programma annuncia il vincitore.

```
void muovi( …..)
{….}

main()
{srand(time(0));              //serve per randomizzare rand();
 int lepre=10, tartaruga=12;
  …
}
```

ES 2 Sviluppare quanto possibile, affrontando i punti in ordine progressivo.

Punto1: Dispensa, pag. 36. Esercizio 1.13.2.
Ci sono diversi modi di svolgere l'esercizio.
E' richiesto di:

- scrivere una funzione min_array che riceve come parametro un array di interi e la sua dimensione, e restituisce il valore minimo contenuto nell'array.

- scrivere una funzione min_mat che data una matrice di int di dimensione dim x 5 e il valore di dim, usa min_array per determinare il valore minimo della matrice.

Punto2 : Dispensa, pag. 36. Esercizio 1.13.3
Nota*:* se diversi elementi hanno il valore minimo, deve essere restituito il riferimento (o il puntatore) al primo di tali elementi.

Suggerimento pratico per i test: stampare la matrice, stampare il valore, ristampare la matrice. Per es.:

```
main(){
int M[][5]={1,4,0,8,0, -1,3,5,7,2, 0,0,-1, 2,3};

//stampaMat(…);
//cout << min_mat(M,3)<< endl;
//stampaMat(…);

//stampaMat(…);
//cout << rif_min(M,3)<< endl;
//stampaMat(…);

//stampaMat(…);
//cout << * pt_min(M,3)<< endl;
//stampaMat(…);
}
```

Punto3: Dispensa, pag. 36. Esercizio 1.13.4

Nota: Per l'esercizio, va benissimo inizializzare la matrice anche in modo esplicito, nel main. Ci sono almeno due modi:

```
int M[][5]={1,4,0,8,0, -1,3,5,7,2, 0,0,-1, 2,3};
int M[][5]={ {1,4,0,8,0}, { -1,3,5,7,2}, {0,0,-1, 2,3} };
```

Punto4 (per i piu' ambiziosi). Pronti per una sfida?

La funzione min_array accetta un array di qualunque dimensione.
Le funzioni sulle matrici che abbiamo sviluppato, invece, accettano solo matrici di dimensione N x 5, dove N puo' variare, ma 5 no.

Riusciete a pensare un modo di riscrivere la funzione di ricerca del minimo in modo tale che accetti matrici di qualunque dimensione?
Insomma, si vorrebbe avere la stessa liberta' che si ha con gli array.

Vorremmo passare alla funzione il puntatore al primo elemento della matrice, ed il numero totale dei suoi elementi.

Suggerimento. Pensare a come la matrice e' implementata: le celle di una matrice (a 2,3,4 dimensioni) sono immagazzinati nella RAM in posizioni contigue.