Es. 7.1 – Creazione vettore sullo heap

Scrivere una funzione *void componenti_negative(…)* che, dato un vettore in ingresso contenente valori interi (dunque sia positivi che negativi o nulli), restituisca un nuovo vettore contenente le sole componenti negative, nello stesso ordine. La funzione deve allocare il nuovo vettore sullo heap. Tale vettore verrà poi deallocato dal chiamante.

Scrivere un programma che applica la funzione *componenti_negative(…)* al seguente vettore:
{11, -22, 4, -3, 18, -1},
stampa a video il vettore risultato, e poi lo distrugge.

Output di esempio:

```
Vettore di partenza (con componenti sia positive che negative):
11 -22 4 -3 18 -1
Ecco il nuovo vettore (solo componenti negative):
-22 -3 -1
Ora che non mi serve piu', posso deallocarlo dallo Heap
```

Es. 7.2 – Somma diagonali

Scrivere una funzione *int somma_diag(const int mat[][3])* che prende in ingresso una matrice quadrata di interi di dimensione 3x3, e restituisce la somma degli elementi sulle diagonali principale e secondaria. L'elemento centrale deve essere sommato solo una volta.
Nota: Realizzare la funzione tramite cicli.

Scrivere un programma che:
- dichiara una matrice 3x3 di numeri interi;
- chiede all'utente di digitare da tastiera 9 numeri interi e li inserisce nella matrice *per righe*;
- invoca la funzione *somma_diag(…)*, passando la matrice come argomento;
- stampa a video il valore restituito dalla funzione.

Output di esempio:

```
Inserisci 9 interi:
1
2
3
4
1
6
7
8
1
La somma degli elementi sulla diag. princ. e sec. e': 13
```

Es. 7.3 – Somma diagonali (versione 2)

Svolgere l'esercizio precedente, ma stavolta la funzione deve accettare matrici quadrate *nxn* di dimensione generica *n*, e deve avere la seguente interfaccia:
int somma_diag(const int mat, int n)*.
L'eventuale elemento centrale deve essere sommato solo una volta.

Output di esempio:

```
Inserisci 9 interi:
1
2
3
4
1
6
7
8
1
La somma degli elementi sulla diag. princ. e sec. e': 13
```

Es. 7.4 – Serpente di numeri

Scrivere una funzione *int* crea_serpente(int r, int c)*. La funzione crea una matrice di dimensione *r*c* fatta nel seguente modo (es. *r*=5, *c*=6):

```
1  2  3  4  5  6
12 11 10 9  8  7
13 14 15 16 17 18
24 23 22 21 20 19
25 26 27 28 29 30
```

La matrice deve essere restituita al chiamante *come valore di ritorno*.

Scrivere un programma che:

- chiede all'utente di inserire da tastiera due interi *N* e *M*;
- invoca la funzione passandole *N* e *M* come argomenti;
- stampa a video il contenuto della matrice restituita dalla funzione.

Nota: Per il corretto spaziamento dei numeri della matrice, usare il carattere tabulazione '\t'.

Output di esempio:

```
Inserisci N: 5
Inserisci M: 6

1       2       3       4       5       6
12      11      10      9       8       7
13      14      15      16      17      18
24      23      22      21      20      19
25      26      27      28      29      30
```

Es. 7.5 – Serpente di numeri (versione 2)

Svolgere l'esercizio precedente, ma stavolta la funzione deve avere la seguente interfaccia: *int** crea_serpente(int r, int c)*.

Es. 7.6 – Matrice trasposta

Scrivere una funzione *int* trasposta(const int* matr, int n)* che, data una matrice di interi *nxn* memorizzata in un unico vettore, restituisce come valore di ritorno la matrice trasposta, cioè la matrice ottenuta scambiando le righe con le colonne. Ad esempio, se la matrice è:

```
1 2 3
4 5 6
7 8 9
```

la trasposta sarà:

```
1 4 7
2 5 8
3 6 9
```

Scrivere un programma che:

- dichiara una matrice di interi 3x3;
- acquisisce da tastiera i 9 elementi di tale matrice, per righe;
- stampa a video tale matrice;
- applica la funzione *trasposta(...)* su tale matrice;
- stampa a video la matrice trasposta.

Output di esempio:

```
Inserisci 9 elementi
1
2
3
4
5
6
7
8
9

Matrice di partenza:
1       2       3
4       5       6
7       8       9

Matrice trasposta:
1       4       7
2       5       8
3       6       9
```

Es. 7.7 – Moltiplicazione matrice-vettore

Scrivere una funzione mulMatVect(...) che prende in ingresso una matrice 3x3 di interi e un vettore di 3 interi e ne restituisce la moltiplicazione matrice x vettore colonna, come valore di ritorno (non come argomento). Un esempio di moltiplicazione matrice x vettore colonna è il seguente:

```
     1 2 3          10
M =  4 5 6      v = 20
     7 8 9          30

       1*10+2*20+3*30       140
M*v =  4*10+5*20+6*30   =   320
       7*10+8*20+9*30       500
```

Scrivere un programma che prende da tastiera i 9 elementi della matrice 3x3, poi prende da tastiera i 3 elementi del vettore, invoca la funzione mulMatVect per calcolarne la moltiplicazione matrice x vettore colonna, e infine stampa il risultato.

Output di esempio:

```
Inserisci i 9 elementi della matrice
1 2 3 4 5 6 7 8 9

Inserisci i 3 elementi del vettore colonna
10
20
30

Risultato:
140
320
500
```

Es. 7.8 – Struttura coda

Realizzare un tipo coda che può contenere un massimo di 10 interi. Realizzare tre funzioni:

- void init(coda& c), che inizializza una coda;
- bool push(coda& c, int v), che inserisce un elemento in coda e restituisce true se l'operazione ha avuto successo, false altrimenti;
- bool pop(coda& c, int& v), che estrae un elemento dalla coda e restituisce true se l'operazione ha avuto successo, false altrimenti.

Realizzare un programma che prende da tastiera 10 interi, e per ognuno lo inserisce in una coda 'pari' se pari o in una coda 'dispari' se dispari. Poi, estrae e stampa tutti gli elementi della coda 'pari' e tutti gli elementi della coda 'dispari'.

Output di esempio:

```
Inserisci 10 numeri:
6 7 1 3 9 5 0 2 4 1
Elementi pari:
6 0 2 4
Elementi dispari:
7 1 3 9 5 1
```