Multidimensional arrays exercises - Taken from pbinfo.ro

Clasa IX-a => Tablouri Bidimensionale(matrice) => Parcurgerea matricelor patratice => Usoare

1. Let's consider a square matrix with N rows and N columns. In this matrix we have 4 areas:
    - 1, the zone which contains the elements which are strictly above the main diagonale and strictly above the second diagonale
    - 2, the zone which contains the elements strictly above the main diagonale and strictly below the second diagonale.
    - 3, the zone which contains the elements strictly below the main diagonale and strictly below the second diagonale.
    - 4, the zone which contains the elements strictly below the main diagonale and strictly above the second diagonale.

- Given a squared matrix and an integer number Z, which representsan area from the matrix, create a program which computes the sum of the elements in the Z area.
    - Solution:

```
    #include <iostream>
    using namespace std;

    int main()
    {
        int mat[4][4]={
                {1,2,3,4},
                {5,6,7,8},
                {9,10,11,12},
                {13,14,15,16}
        };

        int z;
        int sum =0;
        cout<<"Enter the zone for which you want to compute the sum of
the elements: (1-4) ";
        cin >> z;
        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++) {
                if(z == 1 && (j > i && j < 4-i-1)) {
                    sum+= mat[i][j];
                } else if (z == 2 &&  ( j > i && j > (4 -i-1) )) {
                    sum+= mat[i][j];
                } else if (z == 3 && (i > j && j > (4-i-1))) {
                    sum+= mat[i][j];
                } else if (z == 4 && (i > j &&  j < (4-i-1) )) {
                    sum+= mat[i][j];
                }
```

```
            }
        }
        cout<< "The sum of the elements in the zone #"<<z<< " is: " <<
sum;
        return 0;
    }
```

2. Given a 2D array with n rows and n columns which contains natural number, create a C++ program which will compute the absolute difference between the sum of the elements on the main diagonale and the elements on the second diagonale.

    - Solution:

```
    #include <iostream>
    #include <cmath>

    using namespace std;

    int main()
    {
        int mat[4][4]={
                {1,2,3,4},
                {5,6,7,8},
                {9,10,81,12},
                {13,14,15,16}
        };

        int sumMainDiagonale = 0;
        int sumSecondDiagonale = 0;
        for (int i = 0; i < 4; i++ ) {
            for (int j = 0; j < 4; j++) {
                if (i == j) {
                    sumMainDiagonale += mat[i][j];
                } else if (j == (4-1-i)) {
                    sumSecondDiagonale += mat[i][j];
                }
            }
        }

        cout<<"The difference between the main diagonale and the second
diagonale is: " <<  abs(sumMainDiagonale-sumSecondDiagonale);
        return 0;
    }
```

3. Given a 2D array with n rows and n columns, Create a C++ program which will compute the greatest common divisor of the sum of the elements wich are above the main diagonale and the sum of the elements which are below the main diagonale.

    - Solution:

```
#include <iostream>

using namespace std;
int computeGCD(int x, int y);

int main()
{
    int mat[4][4]={
            {1,2,3,4},
            {5,6,7,8},
            {9,10,81,12},
            {13,14,15,16}
    };

    int sumAboveMainDiagonale = 0;
    int sumBelowMainDiagonale = 0;

    for (int i = 0; i < 4; i++ ) {
        for (int j = 0; j < 4; j++) {
            if (i < j) {
                sumAboveMainDiagonale += mat[i][j];
            } else {
                sumBelowMainDiagonale += mat[i][j];
            }
        }
    }

    cout << computeGCD(25, 3);
    return 0;
}

int computeGCD(int x, int y) {
    if(x == 0) {
        return y;
    } else if (y == 0) {
        return x;
    } else {
        while (y > 0) {
            int temp = y;
            y = x % y;
            x = temp;
        }
        return x;
    }
}
```

4. Given a 2D array with n rows and n columns, create a C++ program which will build another matrix which will be symetric with respect to the main diagonale of the given matrix.

    - Solution:

```
#include <iostream>

using namespace std;
int main()
{
    int mat[4][4]={
            {1,2,3,4},
            {5,6,7,8},
            {9,10,81,12},
            {13,14,15,16}
    };


    for (int i = 0; i < 4; i++ ) {
        for (int j = 0; j < 4; j++) {
        mat[i][j] = mat[j][i];
        }
    }

    for (int i = 0; i < 4; i++ ) {
        for (int j = 0; j < 4; j++) {
        cout << mat[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

5. Given a natural number `n`, followed by a square matrix with `n * n` elements, which are natural numbers, create a C++ program which will check if the matrix is a magic square

```
#include <iostream>
bool isMagicSquare(int mat[3][3]);

using namespace std;
int main() {
    int mat[3][3] = {
            {6, 1, 8},
            {7, 5, 3},
            {2, 9, 4}
    };
    if(isMagicSquare(mat)){
        cout << " It is a magic square!";
    } else {
        cout << "It is not a magic square!";
    }
    return 0;
}

bool isMagicSquare(int mat[3][3]) {
```

```
        int mainDiagonaleSum = 0;
        int secondDiagonaleSum = 0;
        for (int i = 0; i < 3; i++) {
            mainDiagonaleSum += mat[i][i];
            secondDiagonaleSum += mat[i][3-1-i];
        }

        if(mainDiagonaleSum != secondDiagonaleSum) {
            return false;
        }

        for(int i = 0; i < 3; i++) {
            int rowSum = 0;
            for (int j = 0; j < 3; j++) {
                rowSum += mat[i][j];
            }
            if(rowSum != mainDiagonaleSum) {
                return false;
            }
        }

        for (int i = 0; i < 3; i++) {
            int colSum = 0;
            for (int j = 0; j < 3; j++){
                colSum += mat[j][i];
            }
            if(colSum != mainDiagonaleSum) {
                return false;
            }
        }

        return true;
    }
```

6. Given a square matrix, create a C++ program that will iterate clockwise the outside of the matrix.

```
    #include <iostream>

    using namespace std;
    int main() {
        const int n = 3;
        int mat[n][n] = {
                {6, 1, 8},
                {7, 5, 3},
                {2, 9, 4}
        };
        for(int i = 0; i < n; i++) {
        cout << mat[0][i] << " ";
        }
```

```
        for (int i = 1; i < n; i++) {
            cout << mat[i][n-1] << " ";
        }

        for (int i = n-2; i >=0;i-- ){
            cout << mat[n-1][i]<<" ";
        }

        for(int i = n-2; i > 0; i--) {
            cout << mat[i][0] <<" ";
        }
        return 0;
    }
```

7. A 2D array representing an area of an ocean containing an iceberg is given; values equal to 1 are part of the iceberg, and those equal to 0 represent water.

It is known that the iceberg is surrounded by water (there is no value of 1 on the edge of the matrix) and that in a certain time interval, all the areas of the iceberg that have at least two sides adjacent to the water will melt.

Determine and display how many time intervals it takes for the iceberg to melt completely. Also, display for each time interval how many ice positions the iceberg has at the beginning of the interval.

```
- Solution:
    ```C++
        #include <iostream>
        const int n = 5;

        using namespace std;
        bool hasAtLeastTwoSidesWithWater(int mat[n][n], int row, int col);
        void meltMarkedCells(int mat[n][n]);

        int main() {
            int mat[n][n] = {
                    {0, 0, 0, 0, 0},
                    {0, 1, 1, 1, 0},
                    {0, 1, 1, 1, 0},
                    {0, 1, 1, 1, 0},
                    {0, 0, 0, 0, 0},
            };

            int icebergCells = 0;
            int interval = 1;
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++){
                    if(mat[i][j] != 0) {
                        icebergCells++;
                    }
                }
```
```

```
            }

            cout<<"Before the #"<<interval<<" interval, the iceberg has " <<
icebergCells << " cells." <<endl;
            while (icebergCells > 0) {
                for (int i = 1; i < n-1;i++) {
                    for (int j = 1; j < n-1; j++) {
                        if( hasAtLeastTwoSidesWithWater(mat, i, j)) {
                            //mark them for melting
                            mat[i][j] = 5;
                            icebergCells--;
                        }
                    }
                }
                meltMarkedCells(mat);
                cout<<"Before the #"<<++interval<<" interval, the iceberg has " <<
icebergCells << " cells." << endl;
            }
            return 0;
        }

        bool hasAtLeastTwoSidesWithWater(int mat[n][n], int row, int col) {
            int sidesWithWater = 0;
            if(mat[row][col] != 0) {
                if(mat[row-1][col] == 0) {
                    sidesWithWater++;
                }
                if(mat[row][col+1] == 0) {
                    sidesWithWater++;
                }
                if(mat[row+1][col] == 0){
                    sidesWithWater++;
                }
                if(mat[row][col-1] == 0) {
                    sidesWithWater++;
                }
            }
            return sidesWithWater >=2;
        }

        void meltMarkedCells(int mat[n][n]) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++){
                    if(mat[i][j] == 5) {
                        mat[i][j] = 0;
                    }
                }
            }
        }
    ```
```
```

8. Create a C++ program which, reading from the input file the characters array `cod(s)`, executes the decrypting operation and displays the initial text `s` whic hhas been coded in the input file.
    - Solution:

```
#include <iostream>
#include <fstream>
using namespace std;

char decode(char c);
int main()
{
    ifstream fin;
    ofstream fout;

    fin.open("in.txt");
    fout.open("out.txt");
    char mat[3][3] = {' '};
    char ch;

    for(int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if(fin>>ch)
            {
                mat[i][j] = ch;
            }
        }
    }

    for(int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            fout << decode(mat[i][j]);
        }
    }
    return 0;
}

char decode(char c)
{
    return c +3;
}
```