Session 10 - Two Dimensional arrays - Deep Dive

- In this session we will go through multiple exercises such that we can get used to the concept of 2D-arrays

Class exercises

1. Write a C++ program which will replace each element from the main diagonale with the average of its neighbors
   - Sample Input:

     ```
     12 13 21 17
     8  9  15 4
     2  3  7  9
     21 24 29 18
     ```

   - Sample Output:

     ```
     10 13 21 17
     8 9 15 4
     2 3 14 9
     21 24 29 19
     ```

   - Solution:

     ```
     #include <iostream>
     using namespace std;
     int computeAverage(int a, int b);
     int computeAverage(int a, int b, int c, int d);
     int main() {

         int matrixA[4][4] = {
                 {12, 13,  21, 17},
                 {8,   9,  15,  4},
                 {2,   3,  7,  9},
                 {21,  24, 29, 18}
         };

         for(int i = 0; i < 4; i++) {
             for (int j = 0; j < 4; j++) {
                 if(i == j) {
                     if(i == 0 && j == 0) {
                         matrixA[i][j] = computeAverage(matrixA[0][1],
     matrixA[1][0]);
                     } else if (i==3 && j == 3) {
     ```

```
                        matrixA[i][j] = computeAverage(matrixA[3][2],
matrixA[2][3]);
                    } else {
                        matrixA[i][j] = computeAverage(matrixA[i-1][j],
matrixA[i][j-1], matrixA[i+1][j], matrixA[i][j+1]);
                    }
                }
            }
        }

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

    int computeAverage(int a, int b) {
        return (a+b) /2;
    }

    int computeAverage(int a, int b, int c, int d) {
        return ( a + b + c + d ) / 4;
    }
```

2. Write a C++ program which will replace the main diagonale with the second diagonale.

   - Sample Input:

     ```
         12 13 21 17
         8  9  15 4
         2  3  7  9
         21 24 29 18
     ```

   - Sample Output:

     ```
         17 13 21 12
         8  15 9  4
         2  7  3  9
         18 24 29 21
     ```

   - Solution:

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

```
int main() {

    int matrixA[4][4] = {
            {12, 13,  21, 17},
            {8,   9,  15,  4},
            {2,   3,  7,  9},
            {21,  24, 29, 18}
    };

    for(int i = 0; i < 4; i++) {
        int temp = matrixA[i][i];
        matrixA[i][i] =  matrixA[i][4 - i-1];
        matrixA[i][4-i-1] = temp;
    }

    for(int i = 0; i < 4; i++) {
        for(int j = 0; j < 4; j++) {
            cout<< matrixA[i][j] << "\t";
        }
        cout << endl;
    }
}
```

3. Write a C++ program which will display the elements from a matrix, which are not present on either diagonale (nor main or second).
   - Sample Input:

     ```
     12 13 21 17
     8  9  15 4
     2  3  7  9
     21 24 29 18
     ```

   - Sample Output:

     ```
     13  21
     8   4
     2   9
     24  29
     ```

   - Solution:

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

     int main() {
     ```

```
    int matrixA[4][4] = {
            {12, 13,  21, 17},
            {8,   9,  15,  4},
            {2,   3,  7,  9},
            {21,  24, 29, 18}
    };

    for(int i = 0; i < 4; i++) {
        for(int j = 0; j < 4; j++) {
            if( i!=j && j != (4 - i -1)) {
                cout << matrixA[i][j] << "\t";
            }
        }
        cout << endl;
    }
}
```

//TODO => Add sample input/output

1. Write a C++ program which will multiply a scalar with a two dimensional matrix
    - The theory says that the result will be a matrix where each element is the element from the first matrix, multiplied with the scalar.
    - Sample Input:

```
Matrix = 2 9 0
         1 3 5
         2 4 7
         8 1 5
Scalar = 4
```

    - Sample Output:

```
8 36 0
4 12 20
8 16 28
32 4 20
```

    - Solution:

```
#include <iostream>
using namespace std;
int main() {

    int matrixA[4][3] = {
            {2, 9, 0},
            {1, 3, 5},
            {2, 4, 7},
            {8, 1, 5}
    };
    int scalar = 4;

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 3; j++) {
            matrixA[i][j] *= scalar;
        }
    }

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 3; j++) {
            cout << matrixA[i][j] << " ";
        }
        cout << endl;
    }
}
```

2. Write a C++ program to multiply 2-dimensional arrays one by the other. This is also called matrix multiplication.

    - Theory:

        - Make sure that the number of columns in the 1st matrix, equals the number of rows in the 2nd matrix
        - Multiply the elements of each row of the first matrix by the elements of each column in the second matrix
        - Add the products as follows, considering the matrix from the example:
            - `( 3*2 + 2*1 + 1*2 + 5*8)` => This will be `result[0][0]`;
            - `( 3*9 + 2*3 + 1*4 + 5*1)` => This will be `result[0][1]`;
            - `( 3*0 + 2*5 + 1*7 + 5*5)` => This will be `result[0][2]`;
            - `( 9*2 + 1*1 + 3*2 + 0*8)` => This will be `result[1][0]`;
            - `( 9*9 + 1*3 + 3*4 + 0*1)` => This will be `result[1][1]`;
            - `( 9*0 + 1*5 + 3*7 + 0*5)` => This will be `result[1][2]`;
        - The resulting matrix has `M rows` X `N columns` where `M` is the number of rows of the first matrix and `N` is the number of columns of the second matrix

    - Sample Input:

```
3 2 1 5    2 9 0
9 1 3 0    1 3 5
           2 4 7
           8 1 5
```

    - Sample Output:

```
    50 42 42
    25 96 26
```

- Solution

```
#include <iostream>
using namespace std;
int main() {
    int matrixA[2][4] = {
        {3,2,1,5},
        {9,1,3,0}
    };

    int matrixB[4][3] = {
        {2,9,0},
        {1,3,5},
        {2,4,7},
        {8,1,5}
    };

    int resultMatrix[2][3];
    for(int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            resultMatrix[i][j] = 0;
            for (int k = 0; k < 4; k++) {
                resultMatrix[i][j] += matrixA[i][k]* matrixB[k][j];
            }
        }
    }

    for(int i = 0; i < 2; i++){
        for (int j = 0; j < 3; j++) {
            cout<<resultMatrix[i][j] << " ";
        }
        cout << endl;
    }
}
```

Homework exercises

1. Write a C++ program which will replace each element from the second diagonale with the average of its neighbors
   - Sample Input:

```
12 13 21 17
8  9  15 4
2  3  7  9
21 24 29 18
```

   - Sample Output:

```
12 13 21 12
8  9  10 4
2  10 7  9
13 24 29 18
```