Lab 3 Specifications

Goals

By the end of this lab you should:

- be more familiar with 2d arrays

Exercise 1 - Matrix Addition

1. Write a function that adds two M x N matrices:
   (declare M and N as global constants where M = 2, N = 3)

```
void matrixAdd( const int a[][N], const int b[][N], int sum[][N] );
```

The addition of two matrices is performed by adding elements in corresponding positions in the matrices. Therefore, matrices that are added must be of the same size; the result is another matrix of the same size.

Example of matrix addition:

$$\begin{bmatrix} 2 & 5 & 1 \\ 0 & 3 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 2 \\ -1 & 4 & -2 \end{bmatrix} = \begin{bmatrix} 3 & 5 & 3 \\ -1 & 7 & -3 \end{bmatrix}$$

2. **Input**: The program should take in two matrices as input from the user in the following form: (User input is **underlined and bolded**)

`Enter Matrix A:` **2 5 1 0 3 -1**

`Enter Matrix B:` **1 0 2 -1 4 -2**

3. **Output**: The program should output the result of the sum of Matrix A and B in the following form:

```
Res:
3 5 3
-1 7 -3
```

Submission Note: Be sure to submit this exercise in a file named ex1.cpp.

Exercise 2 - Matrix Multiplication

1. Write a function that multiplies a `Q x R` matrix by a `R x S` matrix (produces a `Q x S` matrix):

For your final submission to R'Sub, you should declare Q, R, and S as global constants where Q = 2, R = 3, S = 3. However, R'Sub will unit test your function with several different values for Q, R, and S. So you should also test your function on other values for Q, R, and S. Specifically, make sure you test your function when R and S are not the same.

```
void matrixMult(const int a[][R], const int b[][S], int product[][S]);
```

Matrix multiplication is not computed by multiplying corresponding elements of the two matrices. The value in position $c_{i,j}$ of the product C of two matrices A and B is the product of row i of the first matrix and column j of the second matrix:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$

The product of row i and column j requires that the row i and the column j have the same number of elements. Therefore, the first matrix (A) must have the same number of elements in each row as there are in the columns of the second matrix (B). Thus, if A and B both have five rows and five columns, their product has five rows and five columns. Furthermore, for these matrices, we can compute both AB and BA, but in general, they will not be equal.

● Another (non-square) example: if A has two rows and three columns and B has three rows and three columns, the product AB will have two rows and three columns.

Given:

$$A = \begin{bmatrix} 2 & 5 & 1 \\ 0 & 3 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 2 \\ -1 & 4 & -2 \\ 5 & 2 & 1 \end{bmatrix}$$

The first element in the product C = AB is

$$\begin{aligned} c_{1,1} &= \sum_{k=1}^{3} a_{1,k} b_{k,1} \\ &= a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1} \\ &= 2(1) + 5(-1) + 1(5) \\ &= 2. \end{aligned}$$

(Note: `c`$_{1,1}$ will actually be stored in `product[0][0]`.)

Similarly, we can compute the rest of the elements in the product of A and B:

$$AB = C = \begin{bmatrix} 2 & 22 & -5 \\ -8 & 10 & -7 \end{bmatrix}$$

2. **Input**: The program should take in two matrices as input from the user in the following form:

```
Enter Matrix A(size 2x3):  2 5 1 0 3 -1

Enter Matrix B(size 3x3):  1 0 2 -1 4 -2 5 2 1
```

Note: If cin breaks for any reason, all values in the array after that point must be 0. As long as you initialize your arrays to 0 and then read directly into the array, it will work properly.

3. **Output**: The program should output the result of the product of Matrix A and B in the following form:

```
Res:
2 22 -5
-8 10 -7
```

Submission Note: Be sure to submit this exercise in a file named ex2.cpp.