King Saud University **Procedural Programming** **CSC215**
College of Computer and Information Sciences **2023 - Semester 451**
Computer Science Department **LAB 05** **Two-dimensional Arrays**

Exercise 1. Two-dimensional: (4 points)

Create a n-by-m array named `studentGrades`. Each row represents a student, and each column represents a grade on one of the m exams the students took during the semester. The array manipulations are performed by four functions:
Function `minimum`: finds the lowest grade of any student for the semester.
Function `maximum`: finds the highest grade of any student for the semester.
Function `average`: calculates a particular student's semester average.
Function `printArray`: displays the two-dimensional array in a neat, tabular format.

Exercise 2.

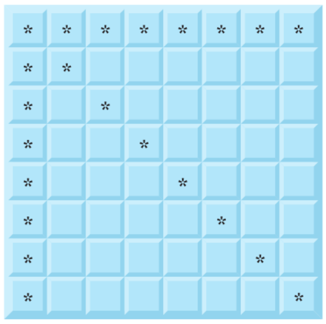
Write a program with a main function to test the function in previous exercises.
Compile and run your program. (1 point)

Lab assignment: Matrix Multiplication: (5 points)

Write a program to calculate the product of 2 matrices, and store the result in a third matrix. You should use functions, header files and prompt the user for the values. (use all what you learned).
Note: An n-by-m two-dimensional matrix can be multiplied by another matrix to give a matrix whose elements are the sum of the products of the elements within a row from the first matrix and the associated elements of a column of the second matrix. Both matrices should either be square matrices, or the number of columns of the first matrix should equal the number of rows of the second matrix.
Hint: To calculate each element of the resultant matrix, multiply the first element of a given row from the first matrix and the first element of a given column in the second matrix, add that to the product of the second element of the same row and the second element of the same column, and keep doing so until the last elements of the row and column have been multiplied and added to the sum.

Bonus exercise: (5 points)

Write a program to solve the "Eight Queens" problem using **backtracking**.
It is simply stated: Is it possible to place eight queens on an empty chess board so that no queen is "attacking" any other—that is, so that no two queens are in the same row, the same column, or along the same diagonal?
Optional: to improve efficiency a heuristic function can be used: It's possible to assign a numeric value to each square of the chessboard indicating how many squares of an empty chessboard are "eliminated" once a queen is placed in that square. For example, each of the four corners would be assigned the value 22, as illustrated in the following diagram:



Once these "elimination numbers" are placed in all 64 squares, an appropriate heuristic might be: Place the next queen in the square with the smallest elimination number. Why is this strategy intuitively appealing?