C++: Practical session 5

1 Heap allocation

Write a program that takes as input a single integer, and then displays Pascal's triangle to that number of rows.

The program should use two heap-allocated arrays to store two rows of the triangle, and include a function with the signature:

```
void calcNextRow(const int* prevRow, int* nextRow, int rowNo);
```

which takes pointers to the previous and current rows and fills the current row using data from the previous row.

This function will be called successively to create all rows of the triangle. You will need to swap the pointers:

```
int* tmpRow = prevRow;
prevRow = nextRow
nextRow = tmpRow;
```

and also output a row of the triangle after each call of the function.

2 Multi-dimensional array heap-allocation

Write separate functions that do the following:

1. Allocate space for an N by N matrix and return a double pointer that can be used to access it.
2. Delete the memory allocated by the first function.

The functions should be usable in the following way:

```
int** matrix = allocateMatrix(3);
int m22 = matrix[2][2];
freeMatrix(matrix);
```

Hint: You will need the construct `int** a = new int*[N]` to allocate an array of pointers to each row.

Write a function that finds the product of two matrices declared in this way. Allow a user to input a size N and then two $N \times N$ matrices and use this function to find their product.

Note that there are (at least) two ways to allocate the matrix memory, either as a single contiguous block, or as separate rows. For practice, you should try both approaches.

2.1 Extensions

1. Use `valgrind` to check that you have freed all the memory that you allocated.

   ```
   valgrind --leak-check=yes --show-reachable=yes ./MyProgram
   ```

2. (Later exercise) When we have covered classes, write an `Array` class that uses a simple block of memory and allows access to it as for a 2D array, with syntax such as

```
Array a(10, 10);
a(9,4) = 7;
a(1,2) = a(8,9) * a(0,0);
```