Objective:

- Creating and manipulating 2D arrays on heap.
- Use of alias and pointers together and const.

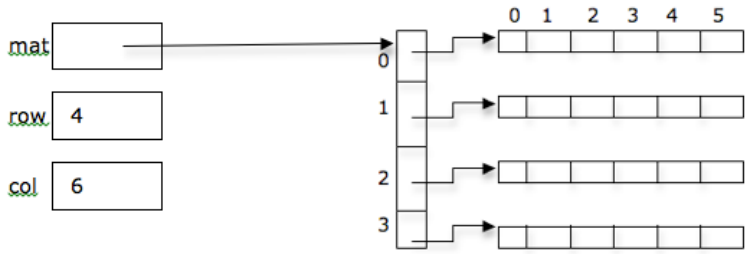
Task: ***Matrices***

In this problem, our goal is to design a library, which will support basic operations of Matrices. The supported operations in this library will be as follows.

Data Structure used for this problem will be as follows:

```
int rows = 4;
int cols = 6;
int * * mat;

mat = new int * [ rows ];
for ( int i=0; i < rows; i = i + 1 )
{
      mat [i] = new int [cols];
}
```



Supported Operations:

1. `void createMatrix (int * * & m, const int row=1, const int Col=1);`

2. `int& at(int * const * const & p, const int r, const int c);`
   For setting or getting some value at a particular location of matrix

3. `void printMatrix(const int * const * const & p, const int rows, const int cols);`

4. `int isIdentity (const int * const * const & p, const int rows, const int cols);`
   if aij = 0 for i != j and aij = 1 for all i = j.

5. `bool isRectangular (const int * const * const & p, const int rows, const int cols);`
   In which number of rows are not equal to number of columns.

6. `bool isDiagonal (const int * const * const & p, const int rows, const int cols);`
   If aij = 0 for all i != j and at least one aij != 0 for i = j;

7. `bool isNullMatrix (const int * const * const & p, const int rows, const int cols);`
   A matrix whose all element are zero.

8. `bool isLowerTriangular (const int * const * const & p, const int rows, const int cols);`

9. `bool isUpperTriangular (const int * const * const & p, const int rows, const int cols);`

10. `bool isTriangular (const int * const * const & p, const int rows, const int cols);`

11. `int * * getMatrixCopy (const int * const * const & p, const int row, const int col );`

12. `bool isEqual(const int * const * const & a, const int row1, const int col1 , const int * const * const & b, const int row2, const int col2);`

13. `void freeMatrix (const int * const * const & p, const int row, const int col);`
    Free the dynamically allocated memory.

14. `int * * transpose (const int * const * const & p, const int row, const int col);`

15. `int * * minus (const int * const * const a, const int rowA, const int colA , const int * const * const b, const int rowB, const int colB);`

16. `void reSize (const int * const * const & p, const int row, const int col, const int newrow, const int newcol );`

17. `bool isSymmetric (const int * const * const & p, const int row, const int col);`
    if $A^t = A$

18. `bool isSkewSymmetric (const int * const * const & p, const int row, const int col);`
    if $A^t = -A$

19. `int * * add (const int * const * const & a, const int row1, const int col1, const int * const * const & b, const int row2, const int col2);`

20. `int * * multiply(const int * const * const & a, const int row1, const int col1 , const int * const * const & b, const int row2, const int col2, int & resultRow, int & resultCol);`