



Objective:

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

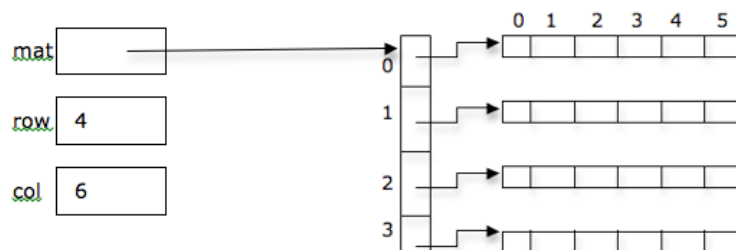
Given to PF – SE-F17

Problem – 1: *Matrix*

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, int row=1, int Col=1);`
2. `int* at(int ** p, int r, int c);`
For setting or getting some value at a particular location of matrix
3. `void printMatrix(int ** p, int rows, int cols)`
4. `int isIdentity (int ** p, int rows, int cols)`
if $a_{ij} = 0$ for $i \neq j$ and $a_{ij} = 1$ for all $i = j$.
5. `bool isLowerTriangular (int ** p, int rows, int cols)`
6. `bool isUpperTriangular (int ** p, int rows, int cols)`
7. `bool isTriangular (int ** p, int rows, int cols)`
8. `int** getMatrixCopy (int ** p, int row, int col)`
9. `bool isEqual(int ** a, int row1, int col1 , int ** b, int row2, int col2)`
10. `void freeMatrix (int *** p , int row, int col);`
Free the dynamically allocated memory.
11. `int** Transpose (int ** p , int row, int col);`
12. `void reSize (int *** p , int row, int col, int newrow, int newcol);`



13. `bool isSymmetric (int ** p , int row, int col)`
`if At = A`
14. `bool isSkewSymmetric (int ** p , int row, int col)`
`if At = -A`
15. `int ** add (int ** a, int row1, int col1 , int ** b, int row2, int col2);`

*IF we apply **Principle of least privilege***

***Use const wherever possible:** Then function prototypes should be as follows:*

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);`
3. `void printMatrix(const int * const * const p, const int rows, const int cols)`
4. `int isIdentity (const int ** p, const int rows, const int cols)`
5. `bool isLowerTriangular (const int * const * const p, const int rows, const int cols);`
6. `bool isUpperTriangular (const int * const * const p, const int rows, const int cols);`
7. `bool isTriangular (const int * const * const p, const int rows, const int cols);`
8. `int** getMatrixCopy (const int * const * const p, const int row, const int col);`
9. `bool isEqual(const int * const * const a, const int row1, const int col1 , const int * const * const b, const int row2, const int col2);`
10. `void freeMatrix (const int * const ** p , const int row, const int col);`
11. `int** Transpose (const int * const * const p , const int row, const int col);`
12. `void reSize (const int * const ** p , const int row, const int col, const int newrow, const int newcol);`
13. `bool isSymmetric (const int * const * const p , const int row, const int col);`
14. `bool isSkewSymmetric (const int * const * const p , const int row, const int col);`
15. `int ** add (const int * const * const a, const int row1, const int col1 , const int * const * const b, const int row2, const int col2);`