* **ITOA**

Write the code for the following function, without using any built-in functions except

malloc or operator new..

char\* itoa(int Value, int Base);

where the returned value is allocated on behalf of the caller, value is the integer to

convert, and base is octal, decimal, or hex.

* [**(6) Spiral Matrix - LeetCode**](https://leetcode.com/problems/spiral-matrix/)
* Modified to return a string

matrix below:

2, 3, 4, 8

5, 7, 9, 12

1, 0, 6, 10

would make the string “2, 3, 4, 8, 12, 10, 6, 0, 1, 5, 7, 9”.

void BuildStringFromMatrix(int\* Matrix, int NumRows, int NumColumns,

char\* OutBuffer)

{

// Your code goes here

}

[(6) Spiral Matrix III - LeetCode](https://leetcode.com/problems/spiral-matrix-iii/)

[(6) Minesweeper - LeetCode](https://leetcode.com/problems/minesweeper/)

[(6) Triangle - LeetCode](https://leetcode.com/problems/triangle/)

[(6) Maximal Square - LeetCode](https://leetcode.com/problems/maximal-square/)

[(6) LRU Cache - LeetCode](https://leetcode.com/problems/lru-cache/)

[(6) Find All Anagrams in a String - LeetCode](https://leetcode.com/problems/find-all-anagrams-in-a-string/)

[(6) Brick Wall - LeetCode](https://leetcode.com/problems/brick-wall/)

[(6) Detect Cycles in 2D Grid - LeetCode](https://leetcode.com/problems/detect-cycles-in-2d-grid/)

[(6) Max Sum of Rectangle No Larger Than K - LeetCode](https://leetcode.com/problems/max-sum-of-rectangle-no-larger-than-k/)

1. Matrix Multiplication

Class Matrix{

Int\* m\_Data;

Constructor dynamically allocates, destructor frees

Matrix multiply(const matrix& other);

}

1. Vector struct

Class Vector{

Int x,y;

Int length();

Int dotProduct(const Vector& other);

};

1. Overload \* operator to multiply matrix, overload +, \* operators for vector
2. Polymorphism (Maybe better for discussion?)

Over-ride member functions so child classes print different values

Entity – print(“I am an entity”)

Robot -print(“Beep boop I am a robot”)

ConfusedCyborg-print(“Greetings, fellow human!”)

Entity\* e = new Entity/Cyborg/Robot