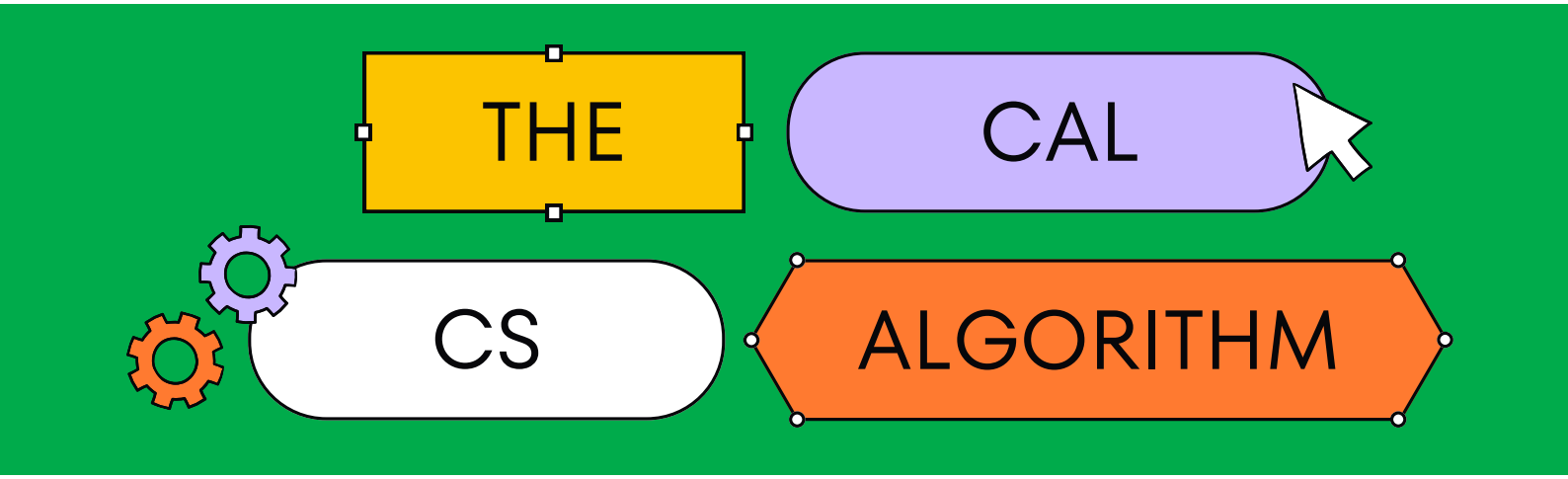

If you have played minesweeper in your childhood days then you may think that it's based on pure luck but it isn't. Surprising! This we realized when planning to create this game that it's not just about the random mines, but more about how many number of mines are associated with a number. If you click on a tile, and the number 2 pops up, then you have 2 mines associated with it. Kaboom! Now you can finally win.

For this, we have developed the "THE CAL CS ALGORITHM". The name tells you all but let's dive into the details of it.

- We have created two loops. The first loop iterates over the rows, while the second loop iterates over the columns. Exciting - but not yet there
- We then checked if the tile was already a mine - using the get_ismine function, which stores the status of each tile on the board. If it was already a mine, it was of no use to us then.
- Then comes the fun part - THE CALCULUS PART(not very fun in real life but let's enjoy it here)
- We have created a variable dx which actually looks at the vertical tiles associated with our original tile. Similarly, we have create a variable dy which looks at the horizontal tiles associated with our tile
- To ensure that there are tiles to the left, right, top, and bottom to our tile, we are checking the boundaries
- If it gets through all the checks, we update the number of mines associated with the tile by using two functions - get_adjacentmines(which is initially set to zero) and set_adjacentmines(which updates the value of the get_adjacentmines)

```
void putadjacentmines()
{
    for (int x = 0; x < rows; x++)
    {
        for (int y = 0; y < columns; y++)
        {
            if (!grid[x][y].get_ismine())
            {
                for (int dx = -1; dx <= 1; dx++)
                {
                    for (int dy = -1; dy <= 1; dy++)
                    {
                        if (x + dx >= 0 && x + dx < rows && y + dy >= 0 && y + dy < columns && !(dx == 0 && dy == 0) && grid[x + dx][y + dy].get_ismine())
                        {
                            grid[x][y].set_adjacentmines(grid[x][y].get_adjacentmines() + 1);
                        }
                    }
                }
            }
        }
    }
}
```