Unit - 1 (Introduction, Asymptotic Notations, Recursive functions)

Ticking Session-3

NAGAVENI L G
PES2UG21CS315
F SEC

1. A matrix is given with nxn dimensions, each row is treated as a binary number and the value of the matrix is given as the sum of these binary numbers. Adarsh wants to maximise this value of the matrix, he is allowed to choose any row or column and toggle each value in that row or column(i.e changing all 0's to 1's, and all 1's to 0's in that row or column). Find the max value of the given matrix after making necessary toggles.

Sample Input 1:

```
3 4              // Order of matrix
0 0 1 1
1 0 1 0          // Elements of matrix
1 1 0 0
```

Sample Output 1:

Output -
39

```
PS C:\Users\Praka\OneDrive\Documents\DAA> gcc week3.c
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
3 4
0 0 1 1
1 0 1 0
1 1 0 0
39
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

```c
#include <stdio.h>
#include <stdlib.h>
#include<math.h>

int matrixScore(int **grid, int gridSize, int *gridColSize)
{
    // Write your code here
    int count = 0;
    for(int i = 0; i < gridSize; i++)
    {
        for(int j = 0; j < *gridColSize; j++)
        {
            if(grid[i][j] == 0)
            {
                grid[i][j] = 1;
                count++;
            }
            else if(count > 0)
            {
                grid[i][j] = 0;
            }
            else
            {
                break;
            }
        }
        count = 0;
    }
    count = 0;

    for(int i = 0; i < *gridColSize; i++)
    {
        int zero_count = 0;
        int one_count = 0;
        for(int j = 0; j < gridSize; j++)
        {
            if(grid[j][i] == 0)
            {
                zero_count++;
            }
            else{
                one_count++;
            }
        }
        for(int j = 0; j < gridSize; j++)
        {
            if(zero_count > one_count)
            {
```

```
                if(grid[j][i] == 0)
                {
                    grid[j][i] = 1;
                }
                else{
                    grid[j][i] = 0;
                }
            }
        }
        count = 0;
        one_count = 0;
        zero_count = 0;
    }

    int sum = 0;
    int c =0;
    for(int i = 0; i < gridSize; i++)
    {
        for(int j = (*gridColSize)-1; j >=0; j--)
        {

            sum = sum + (grid[i][c] * (pow(2,j)));
            c++;
        }
        c = 0;
    }

    return sum;
}

void main()
{
    // Driver Code
    int n;
    int m;
    scanf("%d", &n);

    int **mat = (int **)malloc(n * sizeof(int *));

    scanf("%d", &m);

    for (int i = 0; i < n; i++)
        mat[i] = (int *)malloc(m * sizeof(int));

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
```

```
            scanf("%d", &mat[i][j]);
        }
    }

    printf("%d", matrixScore(mat, n, &m));
}
```

Test cases:

1.

1 1 // Order of matrix

0

Output-
1

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
1 1
0
1
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

2.

2 1 // Order of matrix

0
1

Output-
2

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
2 1
0
1
2
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
```

3.

4 5 // Order of matrix

0 1 1 0 0
1 1 1 1 1
0 0 0 0 0
1 0 1 0 1

Output-
102

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4 5
0 1 1 0 0
1 1 1 1 1
0 0 0 0 0
1 0 1 0 1
102
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

THANK YOU ! ☺