Python - Matrix

1. **A matrix of size m*n is given, write an algorithm to convert even elements to odd by adding ones with them.**

A= [[8,9,12,3],
[3,4,3,2],
[8,9,5,6]]

Algorithm

Step 1: Start
Step 2: Initialize the matrix A of size m*n.
Step 3: For i in range of length of matrix
Step 4: For j in range of length of matrix[i]
Step 5: if matrix [i] [j] % 2==0
Step 6: Set element = element + 1
Step 7: Print matrix A
Step 8: Stop

```
matrix = [[8,9,12,3],[3,4,3,2],[8,9,5,6]]
def convert(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if matrix[i][j] % 2 == 0:
                matrix[i][j] += 1
    return matrix
new_matrix = convert(matrix)
print("New Matrix:")
for row in new_matrix:
    print(row)
```

```
New Matrix:
[9, 9, 13, 3]
[3, 5, 3, 3]
[9, 9, 5, 7]
```

2. **Convert a matrix of size n*m to a new matrix by following the given criteria:**
    a. **a) If an item consists four diagonal elements in its adjacent position sum up all these diagonal elements & replace the elements with the new sum.**

Algorithm

Step 1: Start
Step 2: Input row size (n) and the column size (m) of the matrix A
Step 3: Initialize a matrix A of size n*m.
Step 4: For each row i from 0 to n-1
Step 5: For each column j from 0 to m-1
Step 6: Enter the element A[i][j].
Step 7: Append the row to matrix A.
Step 8: Initialize a new matrix B of size n*m.
Step 9: For each element A[i][j]
Step 10: Calculate the sum of its diagonal neighbors; Sum= A[i-1][j-1] + A[i-1][j+1] + A[i+1][j-1] + A[i+1][j+1]
Step 11: Print matrix B
Step 12: Stop

```python
n = int(input("Enter the number of rows: "))
m = int(input("Enter the number of columns: "))
matrix = []
print("Enter the matrix values one by one:")
for i in range(n):
    row = []
    for j in range(m):
        element = int(input(f"Enter element for row {i+1}, column {j+1}: "))
        row.append(element)
    matrix.append(row)
def diagonal_sum_matrix(matrix):
    n = len(matrix)
    m = len(matrix[0])
    new_matrix = [row[:] for row in matrix]
    for i in range(1, n-1):
        for j in range(1, m-1):
            sum1 = (matrix[i-1][j-1] +  matrix[i-1][j+1] +  matrix[i+1][j-1] + matrix[i+1][j+1])
            new_matrix[i][j] = sum1
    return new_matrix
new_matrix = diagonal_sum_matrix(matrix)
print("\nOriginal Matrix:")
for row in matrix:
    print(row)
print("\nNew Matrix:")
for row in new_matrix:
    print(row)
```

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix values one by one:
Enter element for row 1, column 1: 1
Enter element for row 1, column 2: 4
Enter element for row 1, column 3: 7
Enter element for row 2, column 1: 1
Enter element for row 2, column 2: 6
Enter element for row 2, column 3: 4
Enter element for row 3, column 1: 9
Enter element for row 3, column 2: 3
Enter element for row 3, column 3: 6

Original Matrix:
[1, 4, 7]
[1, 6, 4]
[9, 3, 6]

New Matrix:
[1, 4, 7]
[1, 23, 4]
[9, 3, 6]
```

b. Update the given matrix with size (n+2) and (m+2) rows and column. So, the new matrix may consist 1 in its four borders.

Algorithm

Step 1: Start
Step 2: Initialize a matrix B of size (n+2)*(m+2) and set the values to 1
Step 3: For i from 0 to n-1 do step 4 else step 5
Step 4: For j from 0 to m-1 do step 5 else step 6
Step 5: Copy A[i][j] to B[i+1][j+1]
Step 6: Print B.
Step 7: Stop

```python
[5] matrix = [
        [2,3,1,4],
        [5,14,11,8],
        [9,22,26,3],
        [4,5,6,7]
    ]
    def add_borders(matrix):
        n = len(matrix)
        m = len(matrix[0])
        new_matrix = [[1 for _ in range(m + 2)] for _ in range(n + 2)]
        for i in range(n):
            for j in range(m):
                new_matrix[i + 1][j + 1] = matrix[i][j]
        return new_matrix
    new_matrix = add_borders(matrix)
    print("Original Matrix:")
    for row in matrix:
        print(row)
    print("\nMatrix with Borders of 1:")
    for row in new_matrix:
        print(row)
```

```
Original Matrix:
[2, 3, 1, 4]
[5, 14, 11, 8]
[9, 22, 26, 3]
[4, 5, 6, 7]

Matrix with Borders of 1:
[1, 1, 1, 1, 1, 1]
[1, 2, 3, 1, 4, 1]
[1, 5, 14, 11, 8, 1]
[1, 9, 22, 26, 3, 1]
[1, 4, 5, 6, 7, 1]
[1, 1, 1, 1, 1, 1]
```

c. If the value of n=100 & m=100, use a 3x3 matrix & slide this 3x3 matrix over the 100x100 matrix then if the sum of product of embedded elements are even then replace the 3

Algorithm

Step 1: Start

Step 2: Create a copy of the original matrix.

Step 3: Iterate through each 3x3 submatrix within the original matrix, excluding the last two rows and columns.

Step 4: Calculate the sum of the products of elements in the current submatrix.

Step 5: If the sum is even, replace the corresponding elements in the copy with 0s.

Step 6: If the sum is odd, replace the corresponding elements in the copy with 1s.

Step 7: Return the modified copy

Step 8: Stop

```python
import random
def slide_and_transform(matrix):
    n,m = len(matrix),len(matrix[0])
    new_matrix=[row[:] for row in matrix]
    for i in range(n - 2):
        for j in range(m - 2):
            product_sum=sum(matrix[i+x][j+y] for x in range(3) for y in range(3))
            replacement_value=0 if product_sum%2==0 else 1
            for x in range(3):
                for y in range(3):
                    new_matrix[i + x][j + y] = replacement_value
    return new_matrix
n = 100
m = 100
original_matrix = [[random.randint(1,10) for _ in range(m)]for _ in range(n)]
modified_m=slide_and_transform(original_matrix)
print("Original matrix:")
for row in original_matrix[:5]:
    print(row)
print("\nTransformed matrix:")
for row in modified_m[:5]:
    print(row)
```

```
Original matrix:
[4, 10, 6, 5, 7, 1, 6, 9, 9, 10, 3, 8, 1, 3, 1, 10, 9, 5, 7, 9, 4, 8, 4, 3, 3, 7, 5, 7, 7, 1, 6, 5, 7, 6, 2, 2, 5, 7, 3, 7, 7, 8, 10, 1, 1, 4, 2, 4, 9, 9, 2, 5, 5, 9, 5,
[2, 5, 7, 3, 3, 7, 3, 2, 9, 10, 6, 1, 7, 8, 6, 4, 5, 3, 10, 7, 10, 1, 8, 7, 6, 10, 8, 8, 6, 5, 5, 5, 7, 1, 2, 8, 5, 2, 9, 8, 3, 3, 7, 2, 4, 2, 6, 9, 8, 3, 4, 9, 7, 2, 6,
[10, 7, 7, 6, 6, 4, 2, 6, 1, 1, 10, 9, 5, 5, 8, 10, 6, 4, 5, 5, 4, 2, 5, 5, 10, 8, 1, 8, 7, 9, 1, 6, 8, 4, 6, 3, 5, 3, 8, 10, 5, 9, 7, 3, 10, 10, 8, 4, 5, 9, 6, 10, 9, 1
[9, 10, 2, 10, 10, 2, 5, 9, 6, 10, 7, 6, 6, 2, 2, 4, 6, 4, 9, 6, 3, 9, 10, 4, 6, 3, 10, 8, 1, 1, 10, 5, 5, 7, 10, 1, 3, 9, 8, 9, 10, 5, 6, 2, 1, 1, 6, 4, 1, 2, 3, 4, 8,
[9, 8, 7, 5, 7, 8, 10, 5, 6, 3, 1, 5, 10, 5, 10, 9, 10, 2, 4, 5, 9, 3, 7, 3, 6, 9, 7, 3, 4, 7, 4, 5, 4, 1, 4, 8, 3, 10, 4, 4, 9, 6, 3, 3, 3, 3, 3, 2, 1, 8, 6, 7, 4, 3, 1

Transformed matrix:
[0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
[1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
[1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
[1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
[0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
```