Question Number 4

Use the random Sbox you generated for the following

- Write a code to generate its DDT-LAT in your favorite programming language.
- Submit code in a separate file and show the DDT-LAT in answer script
- What is the maximum differential probability of your Sbox? Mention the transition(s) that lead to that.
- What is the maximum bias your Sbox? Mention the input-output mask(s) that lead to that.
- Now use Sage to generate the DDT-LAT for your Sbox.
- What is the meaning of component function of an Sbox?
- Enumerate the component functions of your Sbox using Sage.
- How would you represent your Sbox as a Boolean function using the component functions? [Hint: There will be four functions each representing one output bit]
- Ref: http://match.stanford.edu/reference/cryptography/sage/crypto/sbox.html

Solution.

Python Code to Generate DDT:

```
#Randomly generated sbox

sbox = [12,15,7,3,5,9,10,2,14,11,6,1,0,4,13,8]

# Creating matrix whose values are 0
ans = [[0 for i in range (16)]for j in range(16)]

# Calculate the DDT by finding input and output differences
for x in range(16):
    for dx in range(16):
        # Compute the output difference for S-box values
        dy = sbox[x] ^ sbox[x ^ dx]
        # Increment the count in DDT table for input-output difference pair (dx, dy)
        ans[dx][dy] += 1

#Printing the matrix
for i in ans:
    print(i)


```

Python Code to Generate LAT:

```
import numpy as np

# Define the S-box
sbox = [12, 15, 7, 3, 5, 9, 10, 2, 14, 11, 6, 1, 0, 4, 13, 8]
```

```
# Define the size of the S-box (4-bit means 16 entries)
n = 4
size = 2**n

# Initialize the LAT (Linear Approximation Table)
lat = np.zeros((size, size), dtype=int)

# Compute the LAT
for a in range(size):
    for b in range(size):
        count = 0

        for x in range(size):

            # Dot product (XOR) between input mask 'a' and input 'x'
            input_mask = bin(a & x).count('1') % 2

            # Dot product (XOR) between output mask 'b' and S-box output 'sbox[x]'
            output_mask = bin(b & sbox[x]).count('1') % 2

            # Check if input_mask == output_mask
            if input_mask == output_mask:
                count += 1
        # Populate the LAT with the biased result

        lat[a, b] = count - (size // 2)

# Print the LAT table
print("Linear Approximation Table (LAT):")
print(lat)
```

DDT and Max Differential Probability

```
Differential Distribution Table:
[16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 2, 4, 4, 0, 2, 2, 0, 0, 0, 2, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 4, 4, 2, 0, 2]
[0, 0, 0, 2, 0, 0, 0, 2, 4, 2, 0, 0, 0, 2, 0, 4]
[0, 2, 0, 0, 0, 0, 2, 0, 0, 4, 0, 2, 0, 2, 2, 2]
[0, 0, 0, 0, 0, 4, 0, 0, 0, 2, 4, 2, 2, 0, 2, 0]
[0, 0, 2, 4, 0, 2, 4, 0, 0, 0, 2, 0, 0, 2, 0, 0]
[0, 2, 2, 0, 0, 2, 6, 0, 0, 0, 0, 0, 0, 0, 4, 0]
[0, 2, 4, 0, 2, 2, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0]
[0, 4, 2, 0, 0, 2, 2, 2, 0, 2, 0, 0, 0, 0, 0, 2]
[0, 2, 0, 0, 0, 0, 2, 0, 4, 2, 4, 0, 0, 0, 2, 0]
[0, 0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 6, 2, 0]
[0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 6, 4, 0, 0, 0]
[0, 0, 0, 0, 2, 0, 0, 2, 2, 0, 0, 2, 0, 0, 4, 4]
[0, 2, 0, 2, 2, 0, 0, 6, 2, 2, 0, 0, 0, 0, 0, 0]
[0, 2, 2, 4, 4, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2]
```

```
    Transition(s) Leading to Maximum Differential Probability:

    Maximum Value Identification: Scanning the matrix, the maximum value is 6.

    Finding Transitions: The value 6 appears at the following positions in the matrix:
        Row 7, Column 6
        Row 12, Column 11

    Therefore, the transitions leading to the maximum differential probability of 6 are:
        Input difference 7 to output difference 6
        Input difference 12 to output difference 11

    In summary, the maximum differential probability of the S-box is 6/16, and the
transitions that lead to this probability are:

        Input difference 7 to output difference 6
        Input difference 12 to output difference 11

    Maximum Differential Probability:

    The Maximum Differential Probability (MDP) is:
    MDP=6/16 =0.375
    MDP=6/16 =0.375

    This is the highest differential probability for the S-box.
    Conclusion:

    The maximum differential probability for the S-box is 0.375, and it is achieved by the
 transitions:

        Input difference 7     Output difference 6
        Input difference 12     Output difference 11.
```

LAT and Max Bias

```
    Linear Approximation Table (LAT):
    [[ 8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
     [ 0  2  0 -2 -4  2  0  2  0 -2  0  2  0  2  4  2]
     [ 0  0  2  2 -2 -2  0  0 -2  2  4  0  0  4 -2  2]
     [ 0  2  2  0  2  0  0  2  2 -4  0 -2  4  2 -2  0]
     [ 0 -2 -4  2 -2 -4  2  0  0 -2  0 -2  2  0  2  0]
     [ 0  4  0  0 -2 -2  2 -2  0  0 -4  0 -2  2 -2 -2]
     [ 0  2 -2  0  0  2 -2  0 -6  0  0 -2  2  0  0 -2]
     [ 0  0  2 -2  0 -4 -2 -2 -2 -2  0  4  2 -2  0  0]
     [ 0 -2 -2  0  0  2 -2 -4  0 -2 -2  0  0  2 -2  4]
     [ 0  0  2  2  0  0 -2 -2  0 -4  2 -2 -4  0  2 -2]
     [ 0 -2  4 -2 -2  0  2  0 -2  0 -2 -4  0 -2  0  2]
     [ 0  0  0  0 -2 -2 -6  2  2  2 -2 -2  0  0  0  0]
     [ 0  0  2  6 -2  2  0  0  0  0 -2  2  2 -2  0  0]
     [ 0 -2  2  0  2  0  0 -2  0  2 -2  0  2  4  4 -2]
     [ 0  4  0  0  0  0  0 -4  2  2  2 -2  2 -2  2  2]
```

```
 [ 0  2  0  2  4 -2  0  2 -2  0 -2  0 -2  0  2  4]]

    Maximum Bias: The maximum absolute value in the LAT is 6 (excluding LAT[0][0]). It occurs in multiple positions:
        LAT[6][8] = -6
        LAT[11][6] = -6
        LAT[12][3] = 6

    Maximum Bias Calculation: The maximum bias can be expressed as a fraction of the total number of inputs:
    Bias=6/16=0.375

    Input-Output Mask(s) Leading to Maximum Bias: The maximum bias of 6 is achieved with the following input-output mask combinations:
        Input mask 6     Output mask 8
        Input mask 11     Output mask 6
        Input mask 12     Output mask 3

    Conclusion:

    The maximum bias of your S-box is 0.375, and it is achieved by the following input-output mask combinations:

        Input mask 6     Output mask 8
        Input mask 11     Output mask 6
        Input mask 12     Output mask 3.
```