In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
def probability_from_matrix(matrix: np.array):
    """
    Calculates probabilities
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
       probabilities - numpy array of probabilities
    """
    probabilities = []
    for i in range(len(matrix[0])):
        probabilities.append(np.sum(matrix[[i]]))
    return np.array(probabilities)
```

In [4]:

```
def probability_of_x_y(matrix: np.array):
    """
    Create a matrix consisting of conditional probabilities
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
       represented_x_y - conditional probability matrix
    """
    represented_x_y = []
    probability_y = probability_from_matrix(matrix)
    for i in range(len(matrix[0])):
        represented_props = []
        for j in range(len(matrix[0])):
            represented_props.append(matrix[i][j] / probability_y[j])
        represented_x_y.append(represented_props)
    represented_x_y = np.asarray(represented_x_y)
    return represented_x_y
```

In [5]:

```
def find_entropy(matrix: np.array):
    """
    Find the entropy of a discrete ensemble of x and y.
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
       result_hx, result_hy - entropy of a discrete ensemble x and y
    """
    sum_hx = 0
    sum_hy = 0
    probabilities = probability_from_matrix(matrix)
    for i in range(len(probabilities)):
        sum_hx += probabilities[i] * np.log2(probabilities[i])
        sum_hy += probabilities[i] * np.log2(probabilities[i])
    result_hx = -1 * sum_hx
    result_hy = -1 * sum_hy
    return result_hx, result_hy
```

In [6]:

```
def find_conditional_entropy(matrix: np.array):
    """
    Find the conditional entropy of H(X|Y) and H (Y|X)
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
        hx_x, hy_y - absolute value of conditional entropy of an ensemble X for a fixed ensemble x and y
    """
    sum_x = 0
    px_y = probability_of_x_y(matrix)
    for i in range(len(matrix[0])):
        represented_sum_y = 0
        for j in range(len(matrix[0])):
            if px_y[i][j] != 0:
                represented_sum_y += matrix[i][j] * np.log2(px_y[i][j])
        sum_x += represented_sum_y
    result_hx_y = -1 * sum_x
    represented_sum_y = 0
    for j in range(len(matrix[0])):
        sum_x = 0
        for i in range(len(matrix[0])):
            if px_y[i][j] != 0:
                sum_x += matrix[i][j] * np.log2(px_y[i][j])
        represented_sum_y += sum_x
    result_hy_x = -1 * represented_sum_y
    return abs(result_hx_y), abs(result_hy_x)
```

In [7]:

```
def get_total_entropy(matrix: np.array):
    """
    This function is implemented to find the total entropy H (X,Y)
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
       total_entropy - total entropy
    """
    sum_x = 0
    for i in range(len(matrix[0])):
        represented_sum_y = 0
        for j in range(len(matrix[0])):
            if matrix[i][j] != 0:
                represented_sum_y += matrix[i][j] * np.log2(matrix[i][j])
        sum_x += represented_sum_y
    total_entropy = -1 * sum_x
    return total_entropy
```

In [8]:

```
def find_total_inform(matrix: np.array):
    """
    Find the total information I (X;Y)
    :arg:
       matrix - numpy matrix of common approach the event
    :return:
       result_sum_x - total information
    """
    result_sum_x = 0
    pxy, py_x, py = matrix, probability_of_x_y(matrix), probability_from_matrix(matrix)
    for i in range(len(pxy[0])):
        represented_sum_y = 0
        for j in range(len(pxy[0])):
            if py_x[i][j] != 0 and py[i] != 0:
                represented_sum_y += pxy[i][j] * np.log2(py_x[i][j] / py[i])
        result_sum_x += represented_sum_y
    return result_sum_x
```

In [32]:

```
prob_matrix = np.loadtxt('input.txt')
output_file = open('output.txt', 'w')

prob_matrix
```

Out[32]:

```
array([[0.1, 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
       [0. , 0.1, 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
       [0. , 0. , 0.1, 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
       [0. , 0. , 0. , 0.1, 0. , 0. , 0. , 0. , 0. , 0. ],
       [0. , 0. , 0. , 0. , 0.1, 0. , 0. , 0. , 0. , 0. ],
       [0. , 0. , 0. , 0. , 0. , 0.1, 0. , 0. , 0. , 0. ],
       [0. , 0. , 0. , 0. , 0. , 0. , 0.1, 0. , 0. , 0. ],
       [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.1, 0. , 0. ],
       [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.1, 0. ],
       [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.1]])
```

In [33]:

```
entropy = find_entropy(prob_matrix)
conditional_entropy = find_conditional_entropy(prob_matrix)

output_file.write(
    f"""H(X)={entropy[0]}
H(Y)={entropy[1]}
H(X|Y)={conditional_entropy[0]}
H(Y|X)={conditional_entropy[1]}
H(X,Y)={get_total_entropy(prob_matrix)}
I(X;Y)={find_total_inform(prob_matrix)}""")

output_file.close()
```

In [35]:

```
output = open('output.txt')
print("result:\n")
print("".join(output.readlines()))
```

```
result:

H(X)=3.321928094887362
H(Y)=3.321928094887362
H(X|Y)=0.0
H(Y|X)=0.0
H(X,Y)=3.321928094887362
I(X;Y)=3.321928094887362
```

In [ ]:

```

```