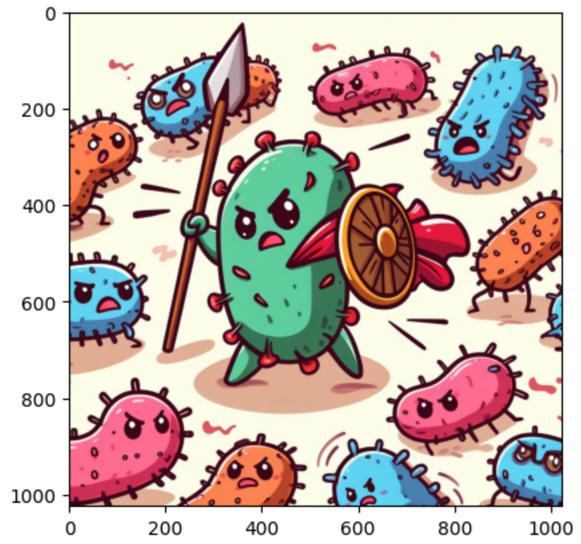
First, load and diplay image from Google Colab. You need to mount your google drive first.

```
#Load and display the image in this cell
import os
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
# the following is the location of the folder that contains '1.png'. Change this to your working folder.
os.chdir('/content/drive/MyDrive/lab3')
# Filename is specified below.
img = plt.imread('1.png')
plt.imshow(img)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=T
<matplotlib.image.AxesImage at 0x7d6d6830d0f0>
```



Task 1. Implement the activation functions shown below (treshold at 0)

> Girintili blok

```
import math
import numpy as np


def step(x):
    output = "Step"
    return 1 if x >= 0 else 0

def sigmoid(x):
    output = 1 / (1 + math.exp(-x))
    return output

def tanh(x):
    output = math.tanh(x)
    return output

def ReLu(x):
    output = max(0, x)
    return output

def LeakyReLu(x): #slope is 0.1 as in the lecture
    output = max(slope * x, x)
    return output


def step(x):
    return np.heaviside(x, 1)

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def tanh(x):
    return np.tanh(x)

def ReLu(x):
    return np.maximum(0, x)

def LeakyReLu(x, slope=0.1):
    return np.maximum(slope * x, x)


print(ReLu(-0.5))
print(tanh(1.5))
print(ReLu(0))
```

```
0
0.9051482536448664
0
```

Task 2. Look back to the image in the beginning and implement the forward pass for the logical model shown in the image. Output the answer. Use Relu for activation

You need to write the code for the forward pass simply by hand. The input is x1=0 and x2=1. The forward pass should be based on the Relu activation function and the activation function should be applied to the final outputs of h1,h2 and y only. Now compute the same forward pass, but for (0,0),(1,0),(1,1)

```
#Put your code below.
def forward_pass(activation_function):
    # Hidden layer
    #h1_in = 1 * x1 + 1 * x2 - 0.5
    h1_in = 1 * 0 + 1 * 1 - 0.5
    h1_out = activation_function(h1_in)

    #h2_in = -1 * x1 + 1 * x2 - 0.5
    h2_in = -1 * 0 + 1 * 1 - 0.5
    h2_out = activation_function(h2_in)

    # Output layer
    y_in = 1 * h1_out + 1 * h2_out - 0.5
    y_out = activation_function(y_in)

    return h1_in, h1_out, h2_in, h2_out, y_in, y_out
#you need to write the input for h1 and h2, the output for h1 and h2. Input for y and output for y.

forward_pass(ReLu)
```

```
(0.5, 0.5, 0.5, 0.5, 0.5, 0.5)
```

```
#Put your code below.
def forward_pass(x1, x2, activation_function):
    # Hidden layer
    h1_in = 1 * x1 + 1 * x2 - 0.5
    h1_out = activation_function(h1_in)

    h2_in = -1 * x1 + 1 * x2 - 0.5
    h2_out = activation_function(h2_in)

    # Output layer
    y_in = 1 * h1_out + 1 * h2_out - 0.5
    y_out = activation_function(y_in)

    return h1_in, h1_out, h2_in, h2_out, y_in, y_out
#you need to write the input for h1 and h2, the output for h1 and h2. Input for y and output for y.

forward_pass(0,0,ReLu)
```

```
(-0.5, 0, -0.5, 0, -0.5, 0)
```

```
forward_pass(1,0,ReLu)
```

```
(0.5, 0.5, -1.5, 0, 0.0, 0)
```

```
forward_pass(1,1,ReLu)
```

```
(1.5, 1.5, -0.5, 0, 1.0, 1.0)
```

Task 3. Now do the same forward pass, but replace Relu with a sigmoid. Output the answer. Is there any difference?

```
#Put your code below.
forward_pass(0,0,sigmoid)
```

```
(-0.5,
 0.3775406687981454,
 -0.5,
 0.3775406687981454,
 0.2550813375962908,
 0.5634267935467275)
```

```
forward_pass(0,1,sigmoid)
```

```
(0.5,
 0.6224593312018546,
 0.5,
 0.6224593312018546,
 0.7449186624037092,
 0.6780704945517833)
```

```
forward_pass(1,0,sigmoid)
```

```
(0.5,
 0.6224593312018546,
 -1.5,
 0.18242552380635635,
 0.30488485500821094,
 0.5756362237654817)
```

```
forward_pass(1,1,sigmoid)
```

```
(1.5,
 0.8175744761936437,
 -0.5,
 0.3775406687981454,
 0.6951151449917892,
 0.667103848561799)
```

Task 4. Calculate the tanh output of a neuron with inputs 0.3 (weighted at 2) and 1 (with a weight of 0.15), and a bias of 0.1

> Girintili blok

```
#Put your code below.
neuron_input = (( 0.3 * 2 + 1 * 0.15 ) + 0.1)
neuron_output = tanh(neuron_input)
print(neuron_output)
```

```
0.6910694698329305
```