Task 1: Implement a primitive neural network

Part 1 and 2: Define the first two layers

In [ ]:

```
import numpy as np

def relu(x):
    return np.maximum(x, 0, x) # this returns 0 if x < 0, otherwise it returns x

def layer(input_vector, weights, bias):
    assert input_vector.shape[0] == weights.shape[1] and weights.shape[0] == bias.s
    return relu(weights @ input_vector + bias)

input_size = 8**2
intermediate_size = 32
output_size = 10

input_vector = np.random.rand(input_size)

weights_1 = np.random.rand(intermediate_size,input_size) # 32 rows x 64 cols
bias_1 = np.random.rand(intermediate_size)

weights_2 = np.random.rand(output_size, intermediate_size) # 10 rows x 32 cols
bias_2 = np.random.rand(output_size)

output = layer(input_vector, weights_1, bias_1)
output = layer(output, weights_2, bias_2)

print(output)
```

```
[269.300292   267.44620465 300.26876959 285.44639683 267.43938116
 272.5996104  274.60207487 250.94251561 272.95960825 275.25058945]
```

Part 3: Generalising to N layers

In [ ]:

```
layers = [64,48,32,10,8,4,4]
input_vector = np.random.rand(layers[0])

def generate_network(layers):
    weights = []
    biases = []
    for index,size in enumerate(layers[1:]):
        m = size; n = layers[index]
        weights.append(np.random.rand(m,n))
        biases.append(np.random.rand(m))
    return weights, biases

def run_network(input_vector, layers, show_dims=False):
    weights, biases = generate_network(layers)
    output = input_vector
    for weight, bias in zip(weights, biases):
        output = layer(output, weight, bias)
        if show_dims:
            print(f"rows: {weight.shape[0]:<3} cols: {weight.shape[1]:<3}")
    return output

run_network(input_vector, layers)
```

Out[ ]:

```
array([173854.75540145, 433757.89059165, 149698.17282257, 293817.82392323])
```

Part 4: Using the generalised function

In [ ]:

```
layers = [64,128,128,128,10]
input_vector = np.random.rand(layers[0])
run_network(input_vector, layers, show_dims=True)
```

```
rows: 128 cols: 64
rows: 128 cols: 128
rows: 128 cols: 128
rows: 10  cols: 128
```

Out[ ]:

```
array([4571957.58559148, 4441769.4188772 , 4114270.31145774,
       4529973.79254949, 4330236.58752122, 4496514.96415654,
       4556754.95187848, 4194672.13774866, 4205138.89445475,
       3965503.24463069])
```