Q 1.2.

a)

1. Import the required libraries, and define the model architecture.
2. Initialize the model and define the loss function.
3. Forward pass, and compute loss.
4. Perform backward pass, and calculate derivatives.
5. Update the parameters.

b)

Layer	Input	Output
Linear1	x	W(1)x + b(1)
f	W(1)x + b(1)	ReLU(W(1)x + b(1))
Linear2	ReLU(W(1)x + b(1))	W(2) ReLU(W(1)x + b(1))+b(2)
g	W(2) ReLU(W(1)x + b(1)) + b(2)	W(2) ReLU(W(1)x + b(1))+b(2)
MSE	(W(2) ReLU(W(1)x + b(1)+b(2)), y)	

c)

Parameter	Gradient
W(1)	δl/δW(1) = δl/δy * δy^/δz3 * δz3/δz2 * δz2/δz1 * δz1/δW(1)
b(1)	δl/δb(1) = δl/δy * δy^/δz3 * δz3/δz2 * δz2/δz1 * δz1/δb(1)
W(2)	δl/δW(2) = δl/δy * δy^/δz3 * δz3/δW(2)
b(2)	δl/δb(2) = δl/δy * δy^/δz3 * δz3/δb(2)

d)

$\delta l/\delta y = 2(y\hat{} - y)$

$\delta y\hat{}/\delta z3 = 1$

$\delta z2/\delta z1 = 0$ if $z1 <= 0$ and 1 if $z1 > 0$

Q 1.3.

a)

Layer	Input	Output
Linear1	x	W(1)x + b(1)
f	W(1)x + b(1)	sigmoid(W(1)x + b(1))
Linear2	sigmoid(W(1)x + b(1))	W(2) sigmoid(W(1)x + b(1))+b(2)
g	Z2 = W(2) sigmoid(W(1)x + b(1)) + b(2)	Sigmoid(Z2)
MSE	Sigmoid(Z2)	

Parameter	Gradient
W(1)	$\delta l/\delta W(1) = \delta l/\delta y * \delta y\hat{}/\delta z3 * \delta z3/\delta z2 * \delta z2/\delta z1 * \delta z1/\delta W(1)$
b(1)	$\delta l/\delta b(1) = \delta l/\delta y * \delta y\hat{}/\delta z3 * \delta z3/\delta z2 * \delta z2/\delta z1 * \delta z1/\delta b(1)$
W(2)	$\delta l/\delta W(2) = \delta l/\delta y * \delta y\hat{}/\delta z3 * \delta z3/\delta W(2)$
b(2)	$\delta l/\delta b(2) = \delta l/\delta y * \delta y\hat{}/\delta z3 * \delta z3/\delta b(2)$

$\delta l/\delta y = 2(y\hat{} - y)$

$\delta y\hat{}/\delta z3 = \sigma(z3)(1-\sigma(z3))$

$\delta z2/\delta z1 = \sigma(z1)(1-\sigma(z1))$

b)

for Q 1.2. b, we need to change the Loss from MSE to BCE.

Also we need to modify the gradient of the loss w.r.t. the predicted output y^:

$\delta bce/\delta y\hat{} = -1/K * (y/y\hat{} - (1-y)/(1-y\hat{}))$

c)

Sigmoid activation function tends to saturate for large positive or negative inputs, resulting in very small gradients. When backpropagating the gradients through multiple layers, the gradients can diminish significantly, leading to the vanishing gradient problem. This can make it difficult for the earlier layers in the network to learn effectively. ReLU, on the other hand, does not suffer from this saturation issue, allowing gradients to flow more freely and mitigate the vanishing gradient problem.