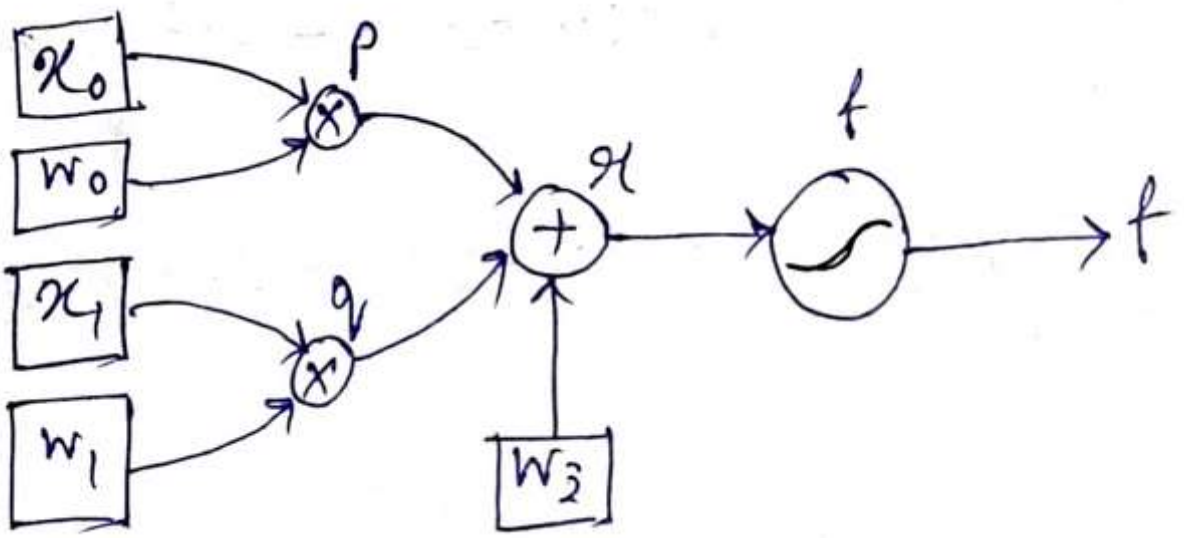
```
import torch
```

Initializing the values as per the question.
Here, the `requires_grad=True` will enable the backward gradient calculation mode.

```
x0 = torch.tensor(-1.0, requires_grad=True)
w0 = torch.tensor(2.0, requires_grad=True)
x1 = torch.tensor(-2.0, requires_grad=True)
w1 = torch.tensor(-3.0, requires_grad=True)
w2 = torch.tensor(-3.0, requires_grad=True)
```

Now, we are adding the internal nodes and operations to our graph as per the below diadram,




```
p = w0 * x0
q = w1 * x1
r = p + q + w2
f = torch.sigmoid(r)
```

Now, we can get value at each stage as.

```
print("Value of p :", p)
print("Value of q :", q)
print("Value of r :", r)
print("Value of f :", f)
```

```
Value of p : tensor(-2., grad_fn=<MulBackward0>)
Value of q : tensor(6., grad_fn=<MulBackward0>)
Value of r : tensor(1., grad_fn=<AddBackward0>)
Value of f : tensor(0.7311, grad_fn=<SigmoidBackward>)
```

To calculate backward gradients, we have to use `backward()` feature

```
f.backward()
```

The gradient wrt. each input are,

```
print("Gradient of x0 :", x0.grad)
print("Gradient of w0 :", w0.grad)
print("Gradient of x1 :", x1.grad)
print("Gradient of w1 :", w1.grad)
print("Gradient of w2 :", w2.grad)
```

```
Gradient of x0 : tensor(0.3932)
Gradient of w0 : tensor(-0.1966)
Gradient of x1 : tensor(-0.5898)
Gradient of w1 : tensor(-0.3932)
Gradient of w2 : tensor(0.1966)
```