Nama : Fadhila Tsani N.A

NIM : 123190133

Plug : Praktikum SCPK E

TUGAS 4 JST

a. Mendefinisikan pola input yang akan diinput :

```
>> p1 = [1;1];
>> p2 = [1;0];
>> p3 = [0;1];
>> p4 = [0;0];
>> input = [p1 p2 p3 p4];
>> input

input =

     1     1     0     0
     1     0     1     0
```

b. Mendefinisikan target :

```
>> t1 = 1;
>> t2 = 1;
>> t3 = 1;
>> t4 = 0;
>> target = [t1 t2 t3 t4];
>> target

target =

     1     1     1     0
```

c. Membuat perceptronnya :

```
>> net = newp([0 1;0 1],1);
>> net

net =

    Neural Network

               name: 'Custom Neural Network'
           userdata: (your custom info)

    dimensions:
```

```
        numInputs: 1
        numLayers: 1
       numOutputs: 1
   numInputDelays: 0
   numLayerDelays: 0
numFeedbackDelays: 0
numWeightElements: 3
       sampleTime: 1

   connections:

      biasConnect: true
     inputConnect: true
     layerConnect: false
    outputConnect: true

   subobjects:

            input: Equivalent to inputs{1}
           output: Equivalent to outputs{1}

           inputs: {1x1 cell array of 1 input}
           layers: {1x1 cell array of 1 layer}
          outputs: {1x1 cell array of 1 output}
           biases: {1x1 cell array of 1 bias}
     inputWeights: {1x1 cell array of 1 weight}
     layerWeights: {1x1 cell array of 0 weights}

   functions:

       adaptFcn: 'adaptwb'
     adaptParam: (none)
       derivFcn: 'defaultderiv'
      divideFcn: (none)
     divideParam: (none)
      divideMode: 'sample'
         initFcn: 'initlay'
     performFcn: 'mae'
   performParam: .regularization, .normalization
       plotFcns: {'plotperform', plottrainstate}
     plotParams: {1x2 cell array of 2 params}
       trainFcn: 'trainc'
     trainParam: .showWindow, .showCommandLine, .show, .epochs,
                 .time, .goal, .max_fail
```

```
    weight and bias values:

                IW: {1x1 cell} containing 1 input weight matrix
                LW: {1x1 cell} containing 0 layer weight matrices
                 b: {1x1 cell} containing 1 bias vector

    methods:

              adapt: Learn while in continuous use
          configure: Configure inputs & outputs
             gensim: Generate Simulink model
               init: Initialize weights & biases
            perform: Calculate performance
                sim: Evaluate network outputs given inputs
              train: Train network with examples
               view: View diagram
        unconfigure: Unconfigure inputs & outputs

    evaluate:       outputs = net(inputs)
```

d. Mendefinisikan bobot awalnya :

```
>> bobot = [-1 1];
>> net.IW{1,1} = bobot;
>> net.IW{1,1}

ans =

    -1     1
```

e. Mendefinisikan nilai bias awalnya :

```
>> bias = [1];
>> net.b{1} = bias;
>> net.b{1}

ans =

     1
```

f. Output dan error pengujian sebelum dilakukan pelatihan perceptron (Output yang ditampilkan hanya hasil keluaran jaringan perceptron berdasarkan bobot dan bias yang sudah dibentuk) :

```
>> output =sim(net,input)

output =

     1     1     1     1
```

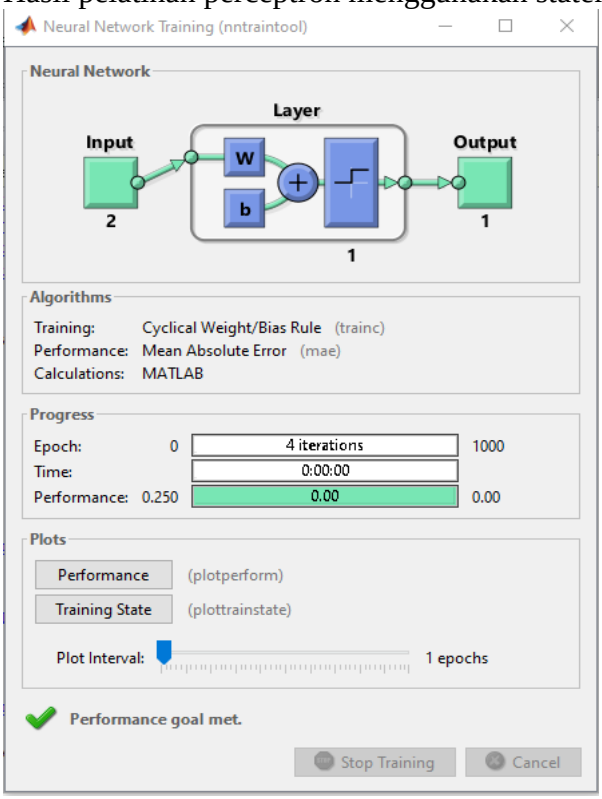
```
>> error = target - output

error =

     0     0     0    -1
```

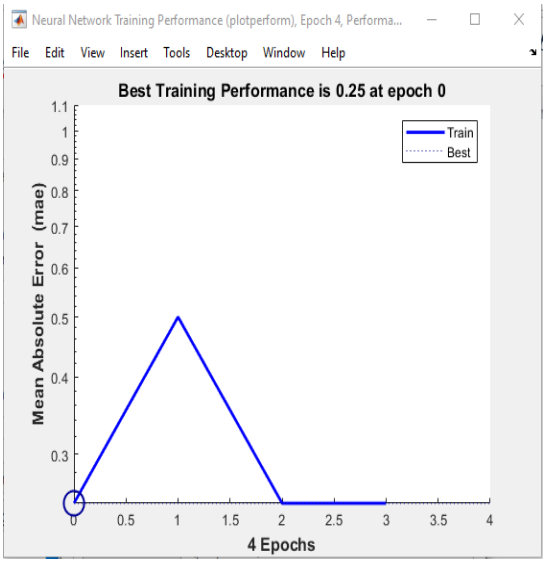
g. Menjalankan pelatihan perceptron :

```
>> net = train(net, input, target)

net =

    Neural Network

              name: 'Custom Neural Network'
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 1
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 3
        sampleTime: 1

    connections:

       biasConnect: true
     inputConnect: true
     layerConnect: false
    outputConnect: true

    subobjects:

             input: Equivalent to inputs{1}
            output: Equivalent to outputs{1}

            inputs: {1x1 cell array of 1 input}
            layers: {1x1 cell array of 1 layer}
           outputs: {1x1 cell array of 1 output}
            biases: {1x1 cell array of 1 bias}
      inputWeights: {1x1 cell array of 1 weight}
      layerWeights: {1x1 cell array of 0 weights}

    functions:
```

```
      adaptFcn: 'adaptwb'
    adaptParam: (none)
      derivFcn: 'defaultderiv'
     divideFcn: (none)
   divideParam: (none)
    divideMode: 'sample'
       initFcn: 'initlay'
    performFcn: 'mae'
  performParam: .regularization, .normalization
      plotFcns: {'plotperform', plottrainstate}
    plotParams: {1x2 cell array of 2 params}
      trainFcn: 'trainc'
    trainParam: .showWindow, .showCommandLine, .show, .epochs,
                .time, .goal, .max_fail

weight and bias values:

           IW: {1x1 cell} containing 1 input weight matrix
           LW: {1x1 cell} containing 0 layer weight matrices
            b: {1x1 cell} containing 1 bias vector

methods:

        adapt: Learn while in continuous use
    configure: Configure inputs & outputs
       gensim: Generate Simulink model
         init: Initialize weights & biases
      perform: Calculate performance
          sim: Evaluate network outputs given inputs
        train: Train network with examples
         view: View diagram
  unconfigure: Unconfigure inputs & outputs

evaluate:       outputs = net(inputs)
```

h. Hasil pelatihan perceptron menggunakan statement train :





i. Menampilkan nilai bobot dan bias yang optimal :

```
>> disp (net.IW{1,1})
     1     1

>> disp (net.b{1})
    -1
```

j. Output dan error pengujian setelah dilakukan pelatihan perceptron :

```
>> output = sim(net,input)

output =

     1     1     1     0

>> error = target - output

error =

     0     0     0     0
```