In your scikit-learn implementation, wet hidden layer sizes to the following values (5,5), (10,10), (30,10) and keep other parameters the same or as default. Report their training and testing accuracy using the same train-test split in the test Part1().

The question ONLY asked us to change the hidden_layer_sizes parameter of the MLPClassifier in the test_SciKit function, no changes were made to our own Neural Network implementation. Since we were asked to set the random state to 1, we are always using the same training set and testing set.

When hidden_layer_sizes = (5,5), here is the output:

```
Accuracy score: 0.4
Confusion Matrix from Part 1b is: [[ 8  0]
 [12  0]]
```

When hidden_layer_sizes = (10, 10), here is the output:

```
Accuracy score: 0.95
Confusion Matrix from Part 1b is: [[ 7  1]
 [ 0 12]]
```

When hidden_layer_sizes = (30, 10), here is the output:

```
Accuracy score: 0.9
Confusion Matrix from Part 1b is: [[ 7  1]
 [ 1 11]]
```

We used the score function of the MLPClassifier to produce the accuracy scores. In general, it can be seen that increasing the number of nodes within the hidden layers results in more accurate prediction. This is due to the increased capacity of the neural network to capture complex patterns in the input data.

However, the accuracy score dropped when the hidden_layer_sizes increased from (10, 10) to (30, 10). This may be the result of overfitting the neural network to the training data, which made it perform worse on the new, unseen, testing data.