Practical 3 –

Code –

```
import numpy as np

class Perceptron:
    def __init__(self, input_size, learning_rate=0.1, epochs=100):
        self.weights = np.zeros(input_size + 1)  # +1 for bias
        self.learning_rate = learning_rate
        self.epochs = epochs
    def activation_function(self, x):
        return 1 if x >= 0 else 0
    def predict(self, x):
        weighted_sum = np.dot(x, self.weights[1:]) + self.weights[0]
        return self.activation_function(weighted_sum)
    def train(self, X, y):
        for epoch in range(self.epochs):
            for i in range(len(X)):
                prediction = self.predict(X[i])
                error = y[i] - prediction
                # Update weights and bias
                self.weights[1:] += self.learning_rate * error * X[i]
                self.weights[0] += self.learning_rate * error



ascii_values = np.array([ord(str(i)) for i in range(10)])


X = np.array([[int(b) for b in f"{bin(val)[2:]:0>8}"] for val in ascii_values])


y = np.array([1 if i % 2 == 0 else 0 for i in range(10)])


perceptron = Perceptron(input_size=X.shape[1])
```

```
perceptron.train(X, y)


print("Digit\tASCII\tPrediction (1 = Even, 0 = Odd)")
for i in range(10):
    prediction = perceptron.predict(X[i])
    print(f"{i}\t{ascii_values[i]}\t{prediction}")


digit = input("Enter a digit (0-9) to check if it's even or odd: ")
if digit.isdigit() and 0 <= int(digit) <= 9:
    ascii_digit = ord(digit)
    binary_features = [int(b) for b in f"{bin(ascii_digit)[2:]:0>8}"]
    prediction = perceptron.predict(binary_features)
    print("Even" if prediction == 1 else "Odd")
else:
    print("Invalid input. Please enter a digit between 0 and 9.")
```

Output –

Digit	ASCII	Prediction (1 = Even, 0 = Odd)
0	48	1
1	49	0
2	50	1
3	51	0
4	52	1
5	53	0
6	54	1
7	55	0
8	56	1
9	57	0

```
Enter a digit (0-9) to check if it's even or odd: 5
Odd
```