```python
import random
import numpy as np
import matplotlib.pyplot as plt

# CONSTANTS
N = 200
BETA = 2
P = 45
T_N = 200000
T_REPEAT = 100


# FUNCTIONS
def generate_random_matrix(n):
    r = np.random.randint(2, size=(n, N))
    r[r == 0] = -1
    return r


def perform_one_trial():
    # x1 = np.linspace(0, T_N - 1, T_N)
    # x2 = np.array([0.] * T_N)
    # fig = plt.figure()
    # ax = fig.add_subplot(111)
    # line1, = ax.plot(x1, x2, 'r-')
    # plt.ion()
    # plt.ylim([0., 1.1])
    # plt.xlim([0., int(T_N / 200) - 1])
    # plt.show()

    # Generate weight matrix
    patterns = generate_random_matrix(P)
    W = (1/N) * np.matmul(patterns.T, patterns)
    np.fill_diagonal(W, 0)

    # Choose first pattern
    nu_index = 1
    chosen_pattern = patterns[nu_index]
    current_state = chosen_pattern.copy()

    order_sum = 0

    for iteration in range(int(T_N / 200)):
        for neuron_index in range(N):
```

```
45
46              # Feed chosen pattern to network
47              selected_weights = np.matrix(W[
   neuron_index]).T
48              b = np.dot(current_state,
   selected_weights).item()
49              # if b == 0:
50              #     b = 1
51
52              b1 = np.exp(-2 * BETA * b)
53
54              p_b = 1 / (1 + b1)
55              new_neuron_value = -1
56              r = random.uniform(0., 1.0)
57
58              if r < p_b:
59                  new_neuron_value = 1
60              current_state[neuron_index] =
   new_neuron_value
61
62          sum_t = 1/N * np.sum(current_state *
   chosen_pattern)
63          order_sum = (iteration*order_sum + sum_t) / (
   iteration + 1)
64      #     x2[iteration] = order_sum
65      #     line1.set_ydata(x2)
66      #     fig.canvas.draw()
67      #     fig.canvas.flush_events()
68      #
69      # input()
70      return order_sum
71
72
73  def perform_n_trials():
74      x1 = np.linspace(0, T_REPEAT - 1, T_REPEAT)
75      x2 = np.array([0.] * T_REPEAT)
76
77      fig = plt.figure()
78      ax = fig.add_subplot(111)
79      line1, = ax.plot(x1, x2, 'r-')
80      plt.ion()
81      plt.ylim([0., 1.1])
82      plt.xlim([0., T_REPEAT - 1])
83      plt.show()
```

```
 84      order_sum = 0.
 85
 86      for i in x1:
 87          a = perform_one_trial()
 88          x2[int(i)] = a
 89          line1.set_ydata(x2)
 90          fig.canvas.draw()
 91          fig.canvas.flush_events()
 92          print(a)
 93          order_sum += a
 94
 95      print("<m1> = " + str(order_sum / float(T_REPEAT
    )))
 96      input() # Block afterwards
 97
 98 # MAIN CODE
 99 perform_n_trials()
100
101
```