```python
# Exercise 2.1 (a) and (b).

import numpy as np
import matplotlib.pyplot as plt

left_index = 0
middle_index = 1
right_index = 2

# Exercise (a), E = 2*k*T
prob_left = 1 / (2 + np.exp(-2))
prob_right = prob_left
prob_middle = np.exp(-2) / (2 + np.exp(-2))

# Exercise (b), E != 2*k*T
E = 10
k = 1
T = 200
prob_left = 1 / (2 + np.exp(-E/(k*T)))
prob_right = prob_left
prob_middle = np.exp(-E/(k*T)) / (2 + np.exp(-E/(k*T)))

no_steps = np.power(10,5) - 1
iterations = np.linspace(1,no_steps+1,no_steps+1)

position = "left"

position_counter = np.array([1,0,0]) # Left, middle, right
transition_counter = np.array([0,0,0]) # Left, middle, right

position_index = 0
transition_index = 1

left_stat = np.zeros((no_steps+1, 2))
right_stat = np.zeros((no_steps+1, 2))
middle_stat = np.zeros((no_steps+1, 2))
left_stat[0, position_index] = 1

for i in range(no_steps):

    # MC simulating new position for the unit
    rand = np.random.uniform()
    if rand < prob_left:
        new_position = "left"
    elif rand < (prob_left + prob_right):
        new_position = "right"
    elif rand < (prob_middle + prob_left + prob_right):
        new_position = "middle"

    # Assigning the unit to the new position if the conditon allows
    if new_position == "left":
        if position == "middle" or position == "left":
            if position == "middle":
                transition_counter[left_index] += 1
            position = new_position
            position_counter[left_index] += 1
        else:
            position = "right"
            position_counter[right_index] += 1

    elif new_position == "right":
        if position == "middle" or position == "right":
            if position == "middle":
                transition_counter[right_index] += 1
            position = new_position
            position_counter[right_index] += 1
        else:
            position = "left"
            position_counter[left_index] += 1

    elif new_position == "middle":
        if position != "middle":
            transition_counter[middle_index] += 1
        position = "middle"
        position_counter[middle_index] += 1

    # Stats for position frequency
    no_left = position_counter[left_index]
    left_stat[i + 1, position_index] = no_left / (i + 2)

    no_right = position_counter[right_index]
    right_stat[i + 1, position_index] = no_right / (i + 2)

    no_middle = position_counter[middle_index]
    middle_stat[i + 1, position_index] = no_middle / (i + 2)

    # Stats for transition frequency
    no_left = transition_counter[left_index]
    left_stat[i + 1, transition_index] = no_left / (i + 2)

    no_right = transition_counter[right_index]
    right_stat[i + 1, transition_index] = no_right / (i + 2)

    no_middle = transition_counter[middle_index]
    middle_stat[i + 1, transition_index] = no_middle / (i + 2)


position_distribution = position_counter / (no_steps + 1)
transition_distribution = transition_counter / (no_steps + 1)

# Output distributions
print(f'Probability distribution of positions: {position_distribution}')
print(f'Probability distribution of transition: {transition_distribution}')

fig, axs = plt.subplots(1,2)

# Plotting position frequency
axs[0].plot(iterations, left_stat[:,position_index], 'o', markersize=2)
axs[0].plot(iterations, right_stat[:,position_index], 'o', markersize=2)
axs[0].plot(iterations, middle_stat[:,position_index], 'o', markersize=2)
axs[0].set_title('Position frequency')
axs[0].set_xlabel('Time step')
axs[0].set_ylabel('Position frequency (no. positions/time step)')
axs[0].legend(['Left', 'Right', 'Middle'])
axs[0].set_box_aspect(1)

axs[1].plot(iterations, left_stat[:,transition_index], 'o', markersize=2)
axs[1].plot(iterations, right_stat[:,transition_index], 'o', markersize=2)
axs[1].plot(iterations, middle_stat[:,transition_index], 'o', markersize=2)
```

```python
axs[1].set_title('Transition frequency')
axs[1].set_xlabel('Time step')
axs[1].set_ylabel('Transition frequency (no. transitions/time step)')
axs[1].legend(['Left', 'Right', 'Middle'])
axs[1].set_box_aspect(1)

fig.suptitle('2.1a: E = 2kT', fontsize=16)
plt.show()
```