Exercise

Authors:

Abate Kevin: **10812892**

Pigato Lorenzo: **10766953** [Team Leader]

EQ2

Consider the following pseudocode for a ESP32-based IoT monitoring system

```
// Global Timer Handle
declare timer_handle as esp_timer_handle_t

// Initialization
function setup_camera():
    initialize_camera(QVGA)

function app_main():
    call setup_camera()
    call setup_timer()
    loop forever:
        delay(100 ms)

function setup_timer():
    declare timer_config as esp_timer_create_args_t
    set timer_config.callback to process_frame
    set timer_config.name to "10_sec_timer"
    call esp_timer_create(&timer_config,
    &timer_handle)
    call esp_timer_start_periodic(timer_handle,10_000_000) // 10s

// Called every 10 seconds
function process_frame(arg):
    image = capture_camera_frame()
    person_count = estimate_number_of_people(image)
    if person_count == 0:
        payload = create_message(size=1KB)
    else if person_count == 1:
        payload = create_message(size=3KB)
    else:
        payload = create_message(size=6KB)
```

Assuptions

- The system is operated with IEEE 802.15.4 in beacon-enabled mode (CFP only)
- The number of people present in the camera frame at any instant follows a Poisson distribution with an average rate of lambda = 0.15 persons/frame

Libraries

```python
import math
```

First Question

> Compute the Probability Mass Function of the output rate of the ESP32 P(r = r_0), P(r = r_1), P(r = r_2), where r_0, r_1 and r_2 are the output rates when there are 0, 1 or more than 1 persons in the captured frame, respectively.

```python
Lambda = 0.15 # person/frame
time_period = 10 #second

message_size_0_person = 1 #KB

message_size_1_person = 3 #KB

message_size_over_2_person = 6 #KB

r0 = 1 / time_period  #KB/s
r1 = 3 / time_period  #KB/s
r2 = 6 / time_period  #KB/s
```

Calculate the **Poisson Probabilities**

```python
results=[]
for x in range(2):
  p_x = (math.exp(-Lambda) * (Lambda**x)) / math.factorial(x)
  results.append(p_x)

for x in range(2):
  print(f"P(r = {x}) = {results[x]}")

p_over_2_person = 1 - (results[0] + results[1])
print(f"P(r >= 2) = {p_over_2_person}")
```

```
P(r = 0) = 0.8607079764250578
P(r = 1) = 0.12910619646375868
P(r >= 2) = 0.010185827111183543
```

Second Question

> Based on the output rate PMF, compute a consistent slot assignment for the CFP in a monitoring system composed of 1 PAN coordinator and 3 camera nodes. Assume nominal bit rate R=250kbps, packets of L=128bytes, 1 packet fits exactly in one slot. Compute Ts (slot time), Number of slots in the CFP, Tactive, Tinactive and the duty cycle of the system.

```python
num_camera_nodes = 3
```

```
num_nodes = num_camera_nodes + 1 # 3 camera nodes + 1 PAN coordinator

nominal_bit_rate_R = 250 * 1000 # bps

packet_size_L = 128 # bytes

bytes_to_bits = 8

Ts = (packet_size_L * bytes_to_bits)*1e3 / nominal_bit_rate_R
print(f"\nTs (slot time): {Ts:.3f} ms")
```

```
Ts (slot time): 4.096 ms
```

```
E_r = (r0 * results[0]) + (r1 * results[1]) + (r2 * p_over_2_person)

print(f"\nExpected output rate E[r]: {E_r:.4f} KB/s per node")
```

```
Expected output rate E[r]: 0.1309 KB/s per node
```

```
min_output_rate_bits_per_sec = r0 * 8 * 1000  # Convert KB/s to bits/s
BI = (packet_size_L * bytes_to_bits) / min_output_rate_bits_per_sec
BI_ms = BI * 1000  # Convert to ms
print(f"BI (Beacon Interval): {BI:.2f} s = {BI_ms:.2f} ms")
```

```
BI (Beacon Interval): 1.28 s = 1280.00 ms
```

```
expected_total_rate = num_camera_nodes * E_r
print(f"Total expected rate for {num_camera_nodes} nodes:
{expected_total_rate:.5f} KB/s")
```

```
Total expected rate for 3 nodes: 0.39274 KB/s
```

```
worst_case_rate_per_node = r2  # 0.6 KB/s per node
packets_per_BI_per_node = math.ceil((worst_case_rate_per_node * BI *
1000) / packet_size_L)
Ncfp = num_camera_nodes * packets_per_BI_per_node
print(f"\nTotal worst-case packets per superframe: {Ncfp} packets")
```

```
Total worst-case packets per superframe: 18 packets
```

```
# Calculate active and inactive times
active_time = (Ncfp + 1) * Ts  # +1 for beacon frame
inactive_time = BI_ms - active_time
print(f"Tactive: {active_time:.3f} ms")
print(f"Tinactive: {inactive_time:.3f} ms")
```

```
Tactive: 77.824 ms
Tinactive: 1202.176 ms
```

```
#The Ncfp solts plus the beacon slot * time slot
duty_cycle = (active_time / BI_ms) * 100
print(f"Duty cycle: {duty_cycle:.2f}%")
```

```
Duty cycle: 6.08%
```

Third Question

How many additional cameras can be added to keep the duty cycle below 10%?

```
max_active_time = BI_ms * 0.1
print(f"\nMax active time for 10% duty cycle: {max_active_time:.3f}
ms")
```

```

Max active time for 10% duty cycle: 128.000 ms
```

```
max_slots = math.floor(max_active_time / Ts) - 1  # -1 for beacon
frame
print(f"Maximum available slots: {max_slots}")
```

```
Maximum available slots: 30
```

```
max_cameras = math.floor(max_slots / packets_per_BI_per_node)
print(f"Maximum number of cameras: {max_cameras}")
```

```
Maximum number of cameras: 5
```

```
additional_cameras = max_cameras - num_camera_nodes
print(f"Additional cameras that can be added: {additional_cameras}")
```

```
Additional cameras that can be added: 2
```

```
new_total_cameras = num_camera_nodes + additional_cameras
new_total_packets = new_total_cameras * packets_per_BI_per_node
new_active_time = (new_total_packets + 1) * Ts  # +1 for beacon frame
new_duty_cycle = (new_active_time / BI_ms) * 100
print(f"New duty cycle with {new_total_cameras} cameras:
{new_duty_cycle:.2f}%")
```

```
New duty cycle with 5 cameras: 9.92%
```

```
verify_cameras = new_total_cameras + 1
verify_total_packets = verify_cameras * packets_per_BI_per_node
verify_active_time = (verify_total_packets + 1) * Ts  # +1 for beacon
frame
verify_duty_cycle = (verify_active_time / BI_ms) * 100
print(f"Duty cycle with {verify_cameras} cameras:
{verify_duty_cycle:.2f}%")
```

```
Duty cycle with 6 cameras: 11.84%
```