Problem 1: Smart Parking Lot Management System

Manages vehicle entry and exit, counts parked vehicles, and checks capacity.

```
def smart_parking(capacity, logs):
    current = 0
    peak = 0

    for log in logs:
        if log == "IN":
            current += 1
            peak = max(peak, current)
        elif log == "OUT" and current > 0:
            current -= 1

    if current > capacity:
        status = "Full"
    else:
        status = "Available"

    return current, peak, status


capacity = 50
logs = ["IN", "IN", "IN", "OUT", "IN", "IN", "OUT"]

parked, peak, status = smart_parking(capacity, logs)
print("Currently Parked Vehicles:", parked)
print("Parking Status:", status)
```

Problem 2: Online Food Delivery Time Estimator

Estimates delivery time using distance, traffic, and weather conditions.

```
def estimate_delivery_time(distance, traffic, weather):
    time = distance * 5

    if traffic == "High":
        time += 15
    elif traffic == "Medium":
        time += 8

    if weather == "Rainy":
        time += 10

    return time


eta = estimate_delivery_time(8, "High", "Rainy")
print("Estimated Delivery Time:", eta, "minutes")
```

Problem 3: Movie Theatre Seat Occupancy Analyzer

Calculates occupancy percentage and determines show status.

```
def seat_occupancy(total_seats, booked_seats):
    booked = len(booked_seats)
    occupancy = (booked / total_seats) * 100

    if occupancy == 100:
        status = "Housefull"
    elif occupancy >= 70:
        status = "Almost Full"
    else:
        status = "Available"

    return occupancy, status


seats = [1] * 150
occ, status = seat_occupancy(200, seats)
print("Occupancy:", int(occ), "%")
print("Show Status:", status)
```

Problem 4: Cloud Server Load Classification System

Calculates average CPU load and classifies server status.

```
def server_load(cpu_readings):
    avg = sum(cpu_readings) / len(cpu_readings)

    if avg < 50:
        status = "Normal"
    elif avg <= 80:
        status = "Warning"
    else:
        status = "Critical"

    return avg, status


avg, status = server_load([45, 60, 70, 85, 90])
print("Average CPU Load:", int(avg), "%")
print("Server Status:", status)
```

Problem 5: Smart Classroom Resource Usage Monitor

Identifies overused classroom resources.

```
def resource_monitor(usage):
    overused = []

    for resource, hours in usage.items():
        if hours > 8:
            overused.append(resource)
```

```
    alert = "Yes" if overused else "No"
    return overused, alert


usage = {
    "Projector": 6,
    "AC": 9,
    "Lights": 4
}

overused, alert = resource_monitor(usage)
print("Overused Resources:", overused)
print("Energy Alert:", alert)
```

Problem 6: Online Event Registration Capacity Controller

Manages registrations, prevents overbooking, and creates waitlist.

```
def event_registration(capacity, registrations):
    confirmed = min(capacity, registrations)
    waitlisted = max(0, registrations - capacity)

    if registrations >= capacity:
        status = "Closed"
    else:
        status = "Open"

    return confirmed, waitlisted, status


confirmed, waitlisted, status = event_registration(100, 105)
print("Confirmed Registrations:", confirmed)
print("Waitlisted Users:", waitlisted)
print("Registration Status:", status)
```