Bahria University,
Karachi Campus

LAB EXPERIMENT NO.
_06_

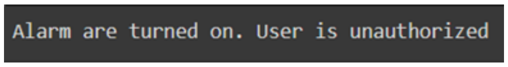
LIST OF TASKS

TASK NO	OBJECTIVE
1.	**Modus Ponens Task:** You're developing a security system for a bank vault. Implement a function that checks if the security camera detects unauthorized access. If the camera detects unauthorized access, trigger the alarm system.
2.	**Modus Tollens Task:** You're creating a temperature monitoring system for a server room. Develop a function that checks if the temperature sensor indicates a temperature above the threshold. If the temperature is not above the threshold, ensure the cooling system remains off.
3.	**Hypothetical Syllogism Task:** You're building a navigation app for drivers. Write a function that determines if the GPS signal is available. If the GPS signal is available, calculate the route to the destination.
4.	**Disjunctive Syllogism Task:** You're developing a scheduling app for students. Implement a function that checks if the user has selected either a morning or evening class. If the user hasn't selected a morning class.
5.	**Simplification Task:** You're developing a game with power up mechanics. Write a function that simplifies the logic for activating a power-up, considering factors such as.
6.	**Addition Task:** You're creating a reservation system for a restaurant. Develop a function that adds a new reservation to the system based on the available time slots and seating capacity.

Submitted On:
4/5/2024

TASK NO 1: Modus Ponens Task:

```
class Unauthorized:
  def __init__(self):
    self.unauth_detected = False
  def detect_unauth(self):
    self.unauth_detected = True
class AlarmController:
  def __init__(self):
    self.alarm_on = False
  def turn_on_alarm (self):
    self.alarm_on = True
def modus_ponens_inference (Unauthorized, alarm_controller):
  if Unauthorized:
    alarm_controller.turn_on_alarm()
u = Unauthorized()
a = AlarmController()
u.detect_unauth()
modus_ponens_inference (u.unauth_detected, a)
if a.alarm_on:
  print("Alarm are turned on. User is unauthorized")
else:
  print("User is authorized. Alarm is not turned on")
```

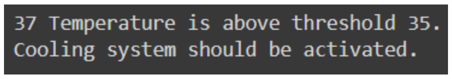


```
Alarm are turned on. User is unauthorized
```

TASK NO 2: Modus Tollens Task:

```
def check_temperature(temp, threshold):
   if temp <= threshold:
      print(f"{temp} Temperature is within threshold {threshold}. \nCooling system remains off.")
   else:
      print(f"{temp} Temperature is above threshold {threshold}. \nCooling system should be activated.")
threshold = 35
temp = 37
check_temperature(temp, threshold)
```

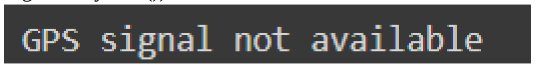
OUTPUT:



```
37 Temperature is above threshold 35.
Cooling system should be activated.
```

TASK NO 3: Hypothetical Syllogism Task:

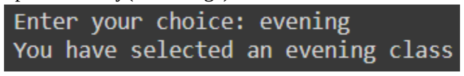
```
def is_gps_signal_available():
   return False
def calculate_route_to_destination():
   return "Route has been calculated..."
def navigate_driver():
   return "Driver has been navigated..."
def navigation_system():
   if is_gps_signal_available():
      route = calculate_route_to_destination()
      result = navigate_driver()
      return route, result
   else:
      return "GPS signal not available"
print(navigation_system())
```



```
GPS signal not available
```

TASK NO 4: Disjunctive Syllogism Task:

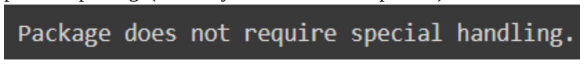
```
def plan_activity (selected_class):
  if selected_class == "morning":
    print("You have selected morning class")
  elif selected_class == "evening":
    print("You have selected an evening class")
  else:
    print("Inavlid selection.")
selected_class = input("Enter your choice: ")
if selected_class != "morning": plan_activity("evening")
else:
  plan_activity("morning")
```



```
Enter your choice: evening
You have selected an evening class
```

TASK NO 5: Simplification Task:

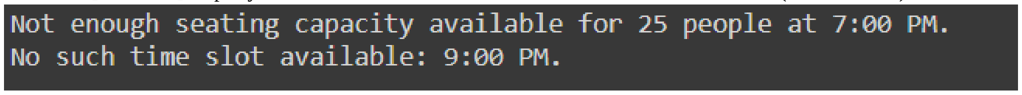
```
def process_package (heavy, destination):
  if heavy and destination:
    print("Package requires special handling.")
  else:
    print("Package does not require special handling.")
is_heavy = False
is_destination_specific = True
process_package(is_heavy, is_destination_specific)
```



```
Package does not require special handling.
```

TASK NO 6: Addition Task:

```
class ReservationSystem:
    def __init__(self):
        self.time_slots = {}
        self.seating_capacity = {}
    def add_time_slot(self, time_slot, capacity):
        self.time_slots[time_slot] = capacity
    def add_reservation(self, time_slot, party_size):
        if time_slot in self.time_slots:
            if self.time_slots[time_slot] >= party_size:
                self.time_slots[time_slot] -= party_size
                print(f"Reservation for {party_size} people at {time_slot}
added successfully!")
            else:   print(f"Not enough seating capacity available for
{party_size} people at {time_slot}.")
        else:       print(f"No such time slot available: {time_slot}.")
restaurant = ReservationSystem()
restaurant.add_time_slot("6:00 PM", 10)
restaurant.add_time_slot("7:00 PM", 15)
restaurant.add_reservation("7:00 PM", 25)
restaurant.add_reservation("9:00 PM", 8)
```



```
Not enough seating capacity available for 25 people at 7:00 PM.
No such time slot available: 9:00 PM.
```