NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

ARTIFICIAL INTELLIGENCE LAB

NAME	Ayesha Imran
Class	CS-A
Lab	09
Course	Artificial Intelligence
Date	31st -December-25
Submitted To	Lec. Ijlal Haider

TASK 01: A box contains 10 white balls, 20 reds and 30 greens. Draw 5 balls with replacement… what is the probability that:

- 3 white or 2 red
- All 5 are the same color

CODE:

```
import math

def nCr(n, r):
    return math.comb(n, r)

p_white = 1/6
p_red = 1/3

# exactly 3 white out of 5
p_3_white = nCr(5, 3) * (p_white**3) * ((1 - p_white)**2)

# exactly 2 red out of 5
p_2_red = nCr(5, 2) * (p_red**2) * ((1 - p_red)**3)

# OR condition
total_probability = p_3_white + p_2_red

print("Probability of 3 white OR 2 red:", total_probability)

#(b) Probability that all 5 are same color
p_green = 1/2

p_all_white = p_white**5
p_all_red = p_red**5
p_all_green = p_green**5

total_same_color = p_all_white + p_all_red + p_all_green

print("Probability all 5 same color:", total_same_color)
```

OUTPUT:

```
PS D:\AI-LAB-13> python -u "d:\AI-LAB-13\Task1.py"
Probability of 3 white OR 2 red: 0.36136831275720177
Probability all 5 same color: 0.035493827160493825
PS D:\AI-LAB-13>
```

TASK 02 : A coin is tossed twice. What is the probability that at least 1 head occurs? The sample space for this experiment is S = {HH, HT, TH, TT}.

CODE:

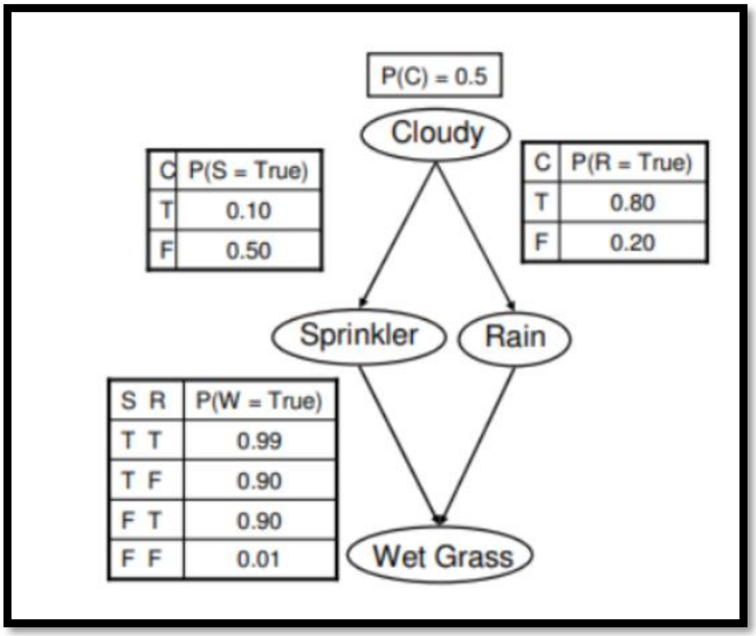
```
probability_at_least_one_head = 1 - (1/4)

print("Probability of at least one head:", probability_at_least_one_head)
```

OUTPUT:

```
Probability of at least one head: 0.75
PS D:\AI-LAB-13>
```

TASK 03 : Predict whether the grass is wet based on whether it is cloudy, raining, or the sprinkler is on.



• For the above a Directed Acyclic Graph, build the Bayesian Network, define appropriate CPTs.

• Calculate the probability of wet grass given that the sprinkler is on and it is raining.

• Calculate the probability of wet grass given that it is cloudy, the sprinkler is off, and it is not raining.

CODE:

```
# Bayesian Network probabilities

# Prior
P_C = 0.5

# Sprinkler given Cloudy
P_S_given_C = {
    True: 0.10,
```

```
    False: 0.50
}

# Rain given Cloudy
P_R_given_C = {
    True: 0.80,
    False: 0.20
}

# Wet Grass given Sprinkler and Rain
P_W_given_S_R = {
    (True, True): 0.99,
    (True, False): 0.90,
    (False, True): 0.90,
    (False, False): 0.01
}

# -------------------------------
# Query 1: P(W = True | S = True, R = True)
# -------------------------------
print("P(Wet Grass | Sprinkler=ON, Rain=TRUE) =",
      P_W_given_S_R[(True, True)])

# -------------------------------
# Query 2: P(W = True | C = True, S = False, R = False)
# (W depends only on S and R)
# -------------------------------
print("P(Wet Grass | Cloudy=TRUE, Sprinkler=OFF, Rain=FALSE) =",
      P_W_given_S_R[(False, False)])
```

Output:

```
PS D:\AI-LAB-13> python -u "d:\AI-LAB-13\Task3.py"
P(Wet Grass | Sprinkler=ON, Rain=TRUE) = 0.99
P(Wet Grass | Cloudy=TRUE, Sprinkler=OFF, Rain=FALSE) = 0.01
PS D:\AI-LAB-13>
```

POST LAB TASKS

TASK 01 : A die is loaded in such a way that an even number is twice as likely to occur as an odd number. If E is the event that a number less than 4 occurs on a single toss of the die, find P (E).

CODE:

```
# Loaded die probabilities

# odd number probability = x
x = 1/9

# probabilities
P = {
    1: x,
    2: 2*x,
    3: x,
    4: 2*x,
    5: x,
    6: 2*x
}

# Event E: number less than 4 -> {1, 2, 3}
P_E = P[1] + P[2] + P[3]

print("P(E) =", P_E)
```

OUTPUT:

```
P(E) = 0.4444444444444444
PS D:\AI-LAB-13>
```

TASK 02 : Let A be the event that an even number turns up and let B be the event that a number divisible by 3 occurs. Find P (A ∪ B) and P (A B).

CODE:

```
# Sample space
S = {1, 2, 3, 4, 5, 6}

# Events
A = {2, 4, 6}       # even numbers
```

```
B = {3, 6}         # divisible by 3

# Probabilities
P_A_intersection_B = len(A & B) / len(S)
P_A_union_B = len(A | B) / len(S)

print("P(A ∩ B) =", P_A_intersection_B)
print("P(A ∪ B) =", P_A_union_B)
```

OUTPUT:

```
PS D:\AI-LAB-13> python -u "d:\AI-LAB-13\Task5.py"
P(A ∩ B) = 0.16666666666666666
P(A ∪ B) = 0.6666666666666666
PS D:\AI-LAB-13>
```

TASK 03 : A manufacturing firm employs three analytical plans for the design and development of a particular product. For cost reasons, all three are used at varying times. In fact, plans 1, 2, and 3 are used for 30%, 20%, and 50% of the products, respectively. The defect rate is different for the three procedures as follows: P(D|P1)=0.01, P(D|P2)=0.03, P(D|P3)=0.02, where P(D|Pj) is the probability of a defective product, given plan j. If a random product was observed and found to be defective, which plan was most likely used and thus responsible.

CODE:

```
# Prior probabilities
P1 = 0.30
P2 = 0.20
P3 = 0.50

# Defect probabilities
D_P1 = 0.01
D_P2 = 0.03
D_P3 = 0.02

# Total probability of defect
P_D = D_P1*P1 + D_P2*P2 + D_P3*P3
```

```python
# Bayes theorem
P_P1_given_D = (D_P1*P1) / P_D
P_P2_given_D = (D_P2*P2) / P_D
P_P3_given_D = (D_P3*P3) / P_D

print("P(P1 | D) =", P_P1_given_D)
print("P(P2 | D) =", P_P2_given_D)
print("P(P3 | D) =", P_P3_given_D)
```

OUTPUT:

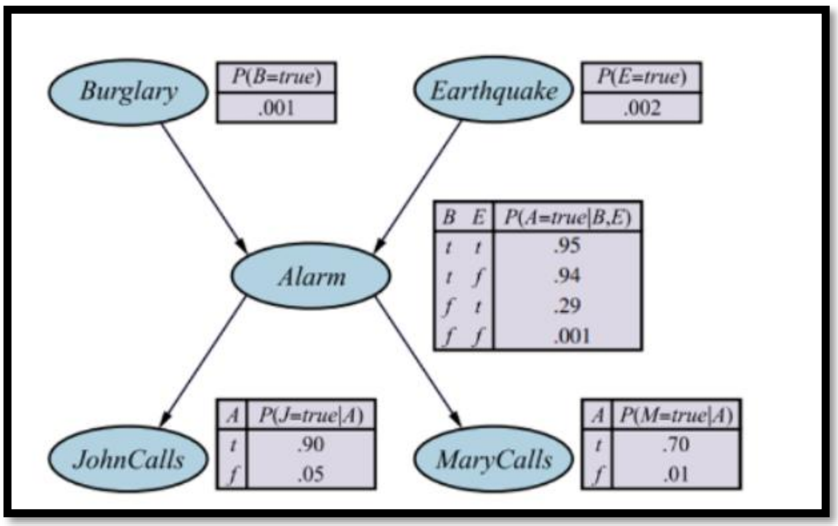
```
PS D:\AI-LAB-13> python -u "d:\AI-LAB-13\Task6.py"
P(P1 | D) = 0.15789473684210525
P(P2 | D) = 0.3157894736842105
P(P3 | D) = 0.5263157894736842
PS D:\AI-LAB-13>
```

TASK 04: Predict whether John or Mary will call given the presence of an alarm, which may be

triggered by a burglary or an earthquake.

• Build the Bayesian Network using the nodes and relationships described above.

• Define appropriate CPTs as shown.



What is the probability that the alarm is triggered given that both burglary and earthquake occur?

What is the probability that John calls if the alarm is triggered?

What is the probability of a burglary given that the alarm is triggered, John calls, and Mary calls?

Explore how the presence of an earthquake affects the alarm probabilities.

CODE:

```python
# Prior probabilities
P_B = 0.001       # Burglary
P_E = 0.002       # Earthquake

# Alarm given Burglary and Earthquake
P_A = {
    (True, True): 0.95,
    (True, False): 0.94,
    (False, True): 0.29,
    (False, False): 0.001
}

# John calls given Alarm
P_J = {
    True: 0.90,
    False: 0.05
}

# Mary calls given Alarm
P_M = {
    True: 0.70,
    False: 0.01
}

# ---------------------------------------------------
# 1) P(Alarm = True | Burglary = True, Earthquake = True)
# ---------------------------------------------------
print("P(A | B, E) =", P_A[(True, True)])

# ---------------------------------------------------
# 2) P(John calls | Alarm = True)
# ---------------------------------------------------
print("P(J | A) =", P_J[True])

# ---------------------------------------------------
# 3) P(Burglary | Alarm, John calls, Mary calls)
# Using Bayes Rule
# ---------------------------------------------------

# P(A, J, M | B)
P_AJM_given_B = (
    P_A[(True, True)] * P_J[True] * P_M[True] * P_E +
    P_A[(True, False)] * P_J[True] * P_M[True] * (1 - P_E)
)
```

```
# P(A, J, M | not B)
P_AJM_given_notB = (
    P_A[(False, True)] * P_J[True] * P_M[True] * P_E +
    P_A[(False, False)] * P_J[True] * P_M[True] * (1 - P_E)
)

# Bayes theorem
P_B_given_AJM = (P_AJM_given_B * P_B) / (
    P_AJM_given_B * P_B + P_AJM_given_notB * (1 - P_B)
)

print("P(Burglary | Alarm, John calls, Mary calls) =", P_B_given_AJM)

# ---------------------------------------------------
# 4) Effect of Earthquake on Alarm
# ---------------------------------------------------
print("P(A | E, no B) =", P_A[(False, True)])
print("P(A | no E, no B) =", P_A[(False, False)])
```

OUTPUT:

```
PS D:\AI-LAB-13> python -u "d:\AI-LAB-13\Task6.py"
P(P1 | D) = 0.15789473684210525
P(P2 | D) = 0.3157894736842105
P(P3 | D) = 0.5263157894736842
PS D:\AI-LAB-13>
```

END