Republic of the Philippines
Laguna State Polytechnic University
Province of Laguna

Machine Problem No. 3			
Topic:	**Topic 2.1: Introduction to Probability in AI**	**Week No.**	5-6
Course Code:	**CSST101**	**Term:**	1st Semester
Course Title:	**Advance Representation and Reasoning**	**Academic Year:**	2024-2025
Student Name		**Section**	
Due date		**Points**	

Machine Problem: Interactive Probability and Decision-Making in AI

Objective:

In this machine problem, you will apply probability theory, Bayesian inference, and decision-making under uncertainty to real-world scenarios using Python. The goal is to create an interactive Python program that allows you to simulate and visualize probabilistic reasoning in various AI contexts.

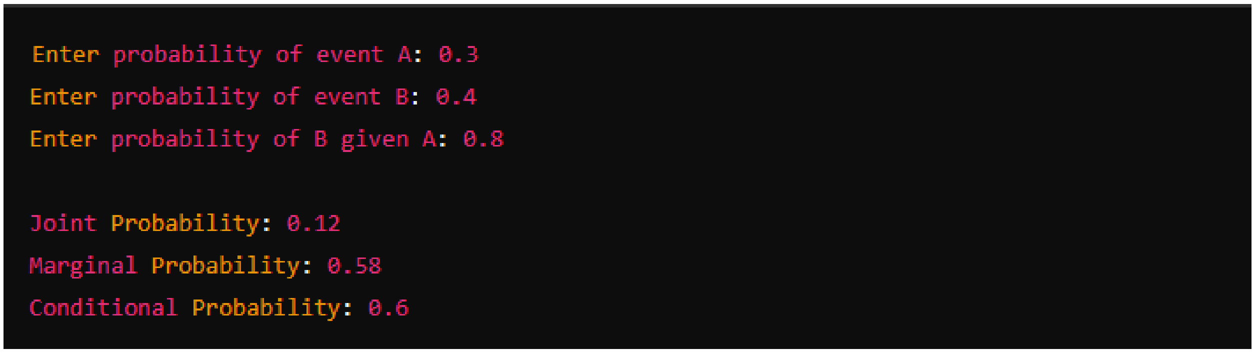
Task Instructions:

Part 1: Implement Basic Probability Calculations

1. **Task:** Write Python functions to calculate the following:
   - **Joint Probability:** The probability of two independent events occurring together.
   - **Marginal Probability:** The probability of a single event by summing over possible joint events.
   - **Conditional Probability:** The probability of an event given the occurrence of another event (using Bayes' Theorem).

2. **Interactive Component:**
   - After writing your functions, the program should prompt the user to input the probabilities of events and calculate the joint, marginal, and conditional probabilities.

- The user will receive feedback based on their inputs and calculations.



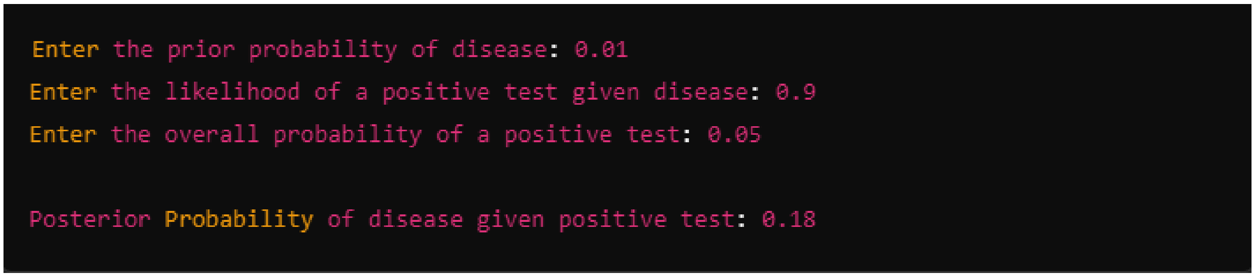
```
Enter probability of event A: 0.3
Enter probability of event B: 0.4
Enter probability of B given A: 0.8

Joint Probability: 0.12
Marginal Probability: 0.58
Conditional Probability: 0.6
```

Part 2: Bayesian Inference for Real-World Scenarios

1. **Task**: Implement a Python function that uses **Bayesian inference** to update probabilities based on new evidence.
    - You are given a medical scenario where you need to update the probability of having a disease based on a positive test result.

2. **Interactive Component**:
    - The program should prompt the user for the prior probability of the disease, the likelihood of a positive test given the disease, and the overall probability of a positive test.
    - Calculate and display the **posterior probability** of having the disease after a positive test result.

Example Interactive Output:



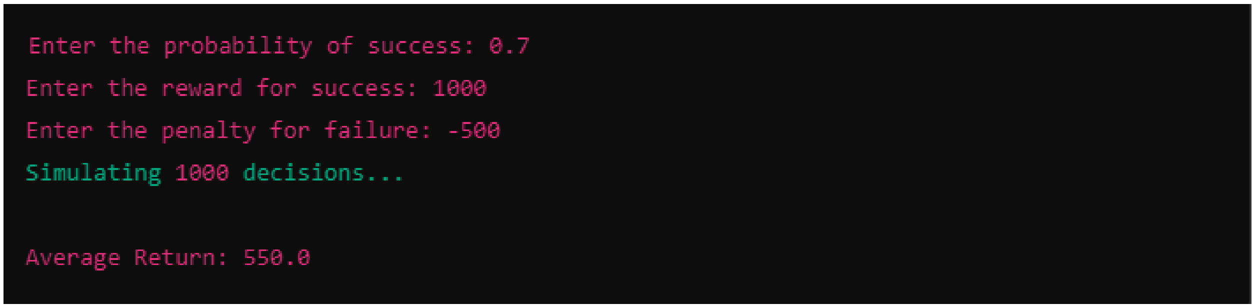
```
Enter the prior probability of disease: 0.01
Enter the likelihood of a positive test given disease: 0.9
Enter the overall probability of a positive test: 0.05

Posterior Probability of disease given positive test: 0.18
```

Part 3: Simulate Decision-Making Under Uncertainty

1. **Task**: Write a Python function to simulate a decision-making process where outcomes are uncertain (e.g., investment decisions). Use a probability distribution to model the decision's potential outcomes.
    - The function should take as inputs the probability of success, reward for success, and penalty for failure.

2. **Interactive Component**:
    - Allow the user to input the probability of success, the reward amount, and the penalty for failure.
    - Run the simulation for a specified number of trials and calculate the **average return**.
    - The user should be able to adjust the probability and observe how it affects the outcome.

Example Interactive Output:



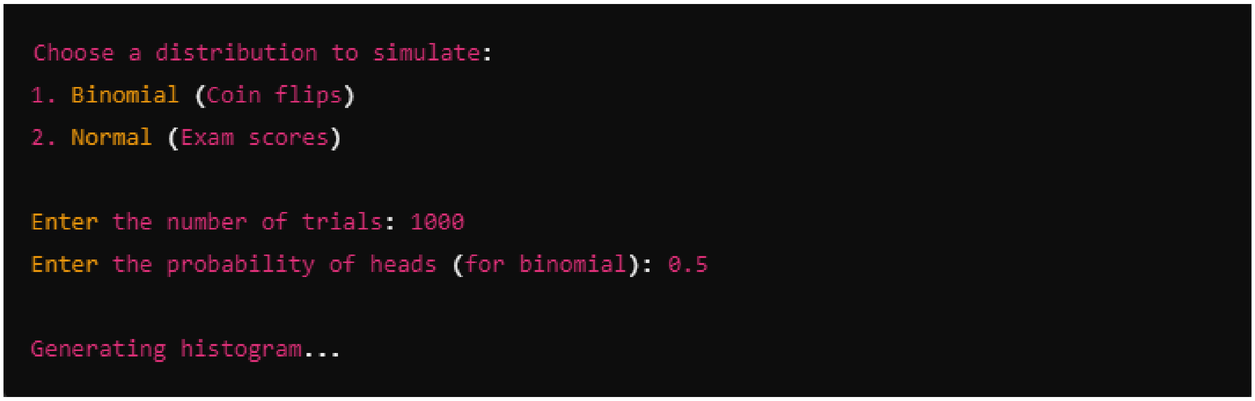
```
Enter the probability of success: 0.7
Enter the reward for success: 1000
Enter the penalty for failure: -500
Simulating 1000 decisions...

Average Return: 550.0
```

Part 4: Visualize Probability Distributions

1. **Task**: Use Python to generate and visualize **binomial** and **normal** probability distributions. The visualizations should represent real-world scenarios like coin flips or dice rolls.

2. **Interactive Component**:
    - Prompt the user to choose between simulating a binomial distribution (e.g., flipping a coin) or a normal distribution (e.g., exam scores).
    - Based on the user's input, generate the distribution and display a histogram using matplotlib.

Example Interactive Output:



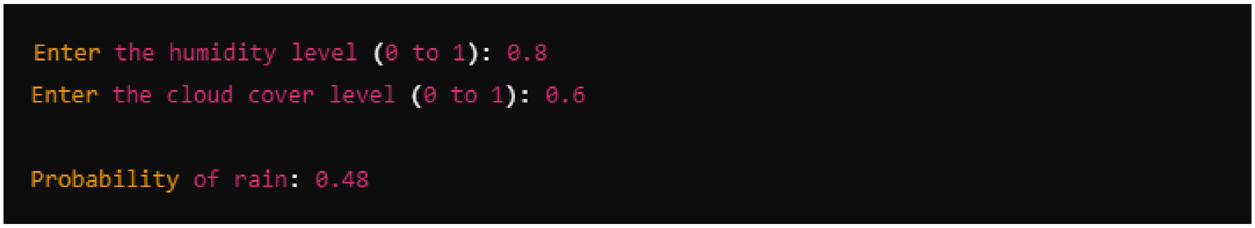
```
Choose a distribution to simulate:
1. Binomial (Coin flips)
2. Normal (Exam scores)

Enter the number of trials: 1000
Enter the probability of heads (for binomial): 0.5

Generating histogram...
```

Part 5: Real-World Scenario Prediction

1. **Task**: Write a Python script that uses **conditional probability** to predict a real-world event. The scenario should involve predicting the probability of rain based on inputs like humidity and cloud cover.

2. **Interactive Component**:
   - Ask the user to input the values for humidity and cloud cover.
   - The script should calculate the probability of rain based on these inputs and provide feedback on how likely it is to rain.

Example Interactive Output:



```
Enter the humidity level (0 to 1): 0.8
Enter the cloud cover level (0 to 1): 0.6

Probability of rain: 0.48
```

Repository Organization:

Folders: Organize your repository with folders such as scripts/, colab_notebooks/, README.md.
Labels: Label files appropriately and maintain clear documentation.

Grading Criteria:

Criteria	Excellent (10 points)	Good (8 points)	Fair (5 points)	Poor (2 points)
Correctness of Implementation	All functions and simulations are correctly implemented, producing accurate results.	Most functions are correctly implemented with minor issues.	Functions are implemented, but contain significant errors.	Incorrect or incomplete implementation of functions.
Interactivity	User interactions are smooth, meaningful, and the program provides accurate feedback.	User interactions are present but could be improved for clarity.	Basic interactivity with limited or unclear feedback.	Poor interactivity or no real-time feedback provided.
Code Quality and Structure	Code is well-organized, easy to read, and includes appropriate comments.	Code is functional but lacks clarity in some areas.	Code runs but is somewhat disorganized or lacks comments.	Code is disorganized and difficult to follow.
Problem-Solving and Application	Demonstrates creative and effective problem-solving with real-world scenario modeling.	Good problem-solving skills with basic real-world scenarios modeled.	Problem-solving approach is basic, and scenarios lack depth.	Little effort in problem-solving, and scenarios are unrealistic.
Documentation and Explanation	Comprehensive, clear, and well-documented code with explanations.	Good documentation with minor gaps.	Limited documentation with significant gaps.	Poor or missing documentation and explanations.

This machine problem encourages the development of an interactive Python program that helps users understand probability theory and decision-making under uncertainty in AI contexts. Through coding, simulations, and real-time feedback, you will demonstrate both technical and conceptual mastery.