Republic of the Philippines
Laguna State Polytechnic University
Province of Laguna

Exercise No. 1			
Topic:	**Topic 1.2: Logic-Based Representation**	**Week No.**	3-4
Course Code:	**CSST101**	**Term:**	1st Semester
Course Title:	**Advance Representation and Reasoning**	**Academic Year:**	2024-2025
Student Name		**Section**	
Due date		**Points**	

Exercise 1: Propositional Logic in Python

Objective:
Students will learn how to represent simple facts and rules using propositional logic, and implement them in Python.

Task:

1. Represent the following statements using propositional logic:
    - "If it rains, the ground will be wet."
    - "If the ground is wet, the match will not light."
    - "It is raining."
2. Implement this using Python to simulate the conditions and use simple logical inference (e.g., using if statements) to determine if the match will light or not.

Instructions:

- Write Python code to evaluate the logical conditions.
- Use basic Boolean expressions to check if the match lights based on the given conditions.

Exercise 2: Predicate Logic Representation

Objective:
Students will apply predicate logic to represent relationships between objects and implement them in Python.

Task:

1. Represent the following scenario using predicate logic:
    - "All humans are mortal."

- "Socrates is a human."
- "Therefore, Socrates is mortal."

2. Implement this in Python using predicates like isHuman() and isMortal() to infer Socrates' mortality.

Instructions:

- Define a predicate for humans (isHuman).
- Use another predicate for mortals (isMortal).
- Write Python code to infer if Socrates is mortal based on these predicates.

Exercise 3: Inference Techniques in Logic-Based Systems

Objective:
Students will develop an inference system to derive conclusions from premises using Python.

Task:

1. Write a Python script that takes a set of facts and rules as input and performs inference using Modus Ponens:
   - Rule: "If X is true, then Y is true."
   - Fact: "X is true."
   - Conclusion: "Y is true."
2. Create a list of rules and facts in Python. Implement a function to apply Modus Ponens and deduce conclusions.

Instructions:

- Implement a system to input multiple rules and facts.
- Write a function that applies Modus Ponens to infer new facts.
- Test your script by providing different rules and facts.

Exercise 4: Hands-on Lab - Implementing a Logic-Based Model in Python

Objective:
Students will implement a logic-based system in Python that performs simple reasoning using propositional or predicate logic.

Task:

1. Create a Python program that models the following scenario:

- "If a person is hungry, they will eat."
- "If a person eats, they will no longer be hungry."
- "John is hungry."

2. Use propositional logic to represent the rules and implement the reasoning process that determines when "**John will no longer be hungry**".

Instructions:

- Define the rules as logical conditions.
- Implement a reasoning system that updates John's hunger status based on the rules.
- Use conditionals and loops in Python to simulate the reasoning process.

Rubric for Exercise 4: Hands-on Lab - Implementing a Logic-Based Model in Python

Criteria	Excellent (90-100%)	Good (75-89%)	Satisfactory (50-74%)	Needs Improvement (0-49%)
Understanding of Logic-Based Models	Demonstrates a thorough understanding of logic-based models and applies them correctly.	Good understanding, with minor mistakes in implementing logic models.	Basic understanding, but significant mistakes in model application.	Lacks understanding of logic-based models, incorrect or incomplete application.
Python Implementation	Code correctly models the scenario and runs without errors.	Code mostly works, with some issues in implementing the logic-based model.	Code runs but contains significant logical errors or model flaws.	Code does not run or fails to model the scenario correctly.
Reasoning Process	Clear and correct implementation of reasoning to determine hunger status.	Reasoning process mostly correct, with minor mistakes or inefficiencies.	Reasoning is implemented but with significant issues or incorrect steps.	Incorrect or missing reasoning process.
Code Structure	Clean, well-organized code with good comments.	Mostly well-organized, with minor issues in readability or comments.	Code is hard to follow and lacks sufficient commenting.	Disorganized code with no comments or clear structure.

Case Study Discussion

Objective:
Analyze a real-world case study where logic-based models are used for decision-making (e.g., in AI systems or expert systems).

Task:

1. Study an AI system that uses logic-based reasoning (e.g., expert systems, chatbots).
2. Discuss how propositional or predicate logic is applied in such systems.
3. Reflect on the advantages and challenges of using logic-based models in real-world applications.

Criteria	Excellent (90-100%)	Good (75-89%)	Satisfactory (50-74%)	Needs Improvement (0-49%)
Understanding of Real-World Applications	Shows deep understanding of how logic-based models are used in real-world applications.	Good understanding with some minor gaps in knowledge.	Basic understanding with significant gaps in real-world applications.	Lacks understanding of real-world applications or provides irrelevant information.
Examples and Discussion	Provides detailed, relevant examples and thoughtful discussion.	Provides good examples, but the discussion lacks depth.	Provides examples, but discussion is limited or incorrect.	Lacks examples or provides irrelevant discussion.
Analysis of Challenges	Thorough analysis of the challenges in applying logic-based systems.	Good analysis with minor gaps in identifying challenges.	Limited analysis with major gaps in understanding challenges.	No meaningful analysis or irrelevant discussion of challenges.
Clarity and Organization	Well-organized and clearly written.	Mostly well-organized, with minor issues in clarity or structure.	Lacks clarity or has poor organization.	Disorganized and unclear discussion.

Assignment: Implement a Logic-Based Model in Python

Objective:
Develop a more complex logic-based system in Python to model real-world scenarios.

Task:

1. Choose a real-world problem that can be modeled using logic (e.g., a simple decision-making system for a chatbot).
2. Write Python code that uses propositional or predicate logic to represent the problem.
3. Implement inference techniques to derive conclusions based on user input or predefined facts.

Instructions:

- Identify the problem and define the logic rules.
- Implement a system that can process input and infer conclusions using logic.
- Submit the Python script as a part of your assignment.

Rubric for Assignment: Implement a Logic-Based Model in Python

Criteria	Excellent (90-100%)	Good (75-89%)	Satisfactory (50-74%)	Needs Improvement (0-49%)
Understanding of Problem	Demonstrates a clear understanding of the problem and logic-based models.	Good understanding, but with minor mistakes in problem-solving approach.	Basic understanding, but significant gaps in problem-solving approach.	Lacks understanding of the problem and incorrect approach.
Python Implementation	Code models the problem effectively, with correct logic and no errors.	Code mostly works, but with minor issues in logic or problem-solving.	Code runs but contains significant errors or does not fully solve the problem.	Code does not run or fails to solve the problem.
Creativity and Complexity	Demonstrates a creative and well-thought-out approach to solving the problem.	Good effort, with minor issues in creativity or complexity of the solution.	Basic solution with limited creativity or complexity.	Solution lacks creativity or complexity, or does not address the problem effectively.
Code Structure	Clean, well-organized code with good comments.	Mostly well-organized code, with minor issues in readability or comments.	Code is hard to follow and lacks sufficient commenting.	Disorganized code with no comments or clear structure.

Lab Work: Logic-Based Reasoning Scripts Submission

Objective:
Students will submit Python scripts that demonstrate logic-based reasoning.

Task:
1. Refine and submit the scripts developed in the previous exercises and assignments.
2. Ensure the scripts demonstrate proper use of propositional and predicate logic, as well as inference techniques.

Instructions:
- Test the Python scripts for accuracy.
- Submit the scripts along with documentation explaining the logic and reasoning process used.

These exercises provide a mix of theory and practical implementation, allowing students to apply logic-based representation and reasoning techniques within Python.

Rubric for Lab Work: Submitting Logic-Based Reasoning Scripts

Criteria	Excellent (90-100%)	Good (75-89%)	Satisfactory (50-74%)	Needs Improvement (0-49%)
Accuracy of Submitted Code	All scripts accurately implement logic-based reasoning and produce correct results.	Most scripts work with minor issues.	Scripts run but contain significant logical errors or missing parts.	Scripts do not run or are incomplete/incorrect.
Code Structure and Documentation	Well-organized and fully documented code.	Code is mostly organized, with minor documentation issues.	Code is poorly organized with insufficient documentation.	Disorganized code with no comments or structure.
Completion of Required Tasks	All exercises and tasks are completed and meet requirements.	Most exercises are completed with minor omissions.	Some exercises are incomplete or missing key components.	Many exercises are missing or incomplete.

Submission Format:
- Upload your Python scripts and Notebook to your GitHub repository.
- Ensure the repository is well-organized, with folders and files clearly labeled (e.g., scripts/, notebooks/, README.md)