****

**Department of Computer Applications Session: 2024- 2025 AI-LAB KCA 351**

**Experiment-No.7**

**Objective:** To implement first order logic

|  |  |  |
| --- | --- | --- |
| **Scheduled Date:** | **Compiled Date:** | **Submitted Date:** |
| 05 Nov 2024 | 05Nov 2024 | 27 Nov2024 |
|  |  |  |
|  |  |  |

|  |
| --- |
| class Term:      def \_\_init\_\_(self, name):          self.name = name      def \_\_repr\_\_(self):          return self.name  class Predicate:      def \_\_init\_\_(self, name, arity):          self.name = name          self.arity = arity      def \_\_call\_\_(self, \*args):          if len(args) != self.arity:              raise ValueError("Invalid number of arguments for predicate")          return Formula(self, args)      def \_\_repr\_\_(self):          return self.name  class Formula:      def \_\_init\_\_(self, predicate, terms):          self.predicate = predicate          self.terms = terms      def \_\_repr\_\_(self):          return f"{self.predicate}({', '.join(map(str, self.terms))})"  class KnowledgeBase:      def \_\_init\_\_(self):          self.statements = []      def add(self, formula):          self.statements.append(formula)      def query(self, formula):          # Simplified reasoning (just checks if the formula is in the knowledge base)          return formula in self.statements  # Example usage  # Define terms  alice = Term("Alice")  bob = Term("Bob")  # Define predicates  likes = Predicate("likes", 2)  # Create a knowledge base  kb = KnowledgeBase()  # Add statements to the knowledge base  kb.add(likes(alice, bob))  # Alice likes Bob  kb.add(likes(bob, alice))  # Bob likes Alice  # Query the knowledge base  query1 = likes(alice, bob)  query2 = likes(bob, alice)  query3 = likes(alice, Term("Charlie"))  print(f"Query 1: {query1}, Result: {kb.query(query1)}")  print(f"Query 2: {query2}, Result: {kb.query(query2)}")  print(f"Query 3: {query3}, Result: {kb.query(query3)}") |

**Department of Computer Applications Session: 2024- 2025 AI-LAB KCA 351**

