**EXP NO. 6 IMPLEMENTATION OF MINIMAX ALGORITHM FOR AN APPLICATION**

AIM:

ALGORITHM:

**PROGRAM:**

import math

def minimax(board, depth, is\_maximizing\_player):

if depth == 0 or game\_over(board):

return evaluate\_board(board)

if is\_maximizing\_player:

max\_eval = -math.inf

for move in possible\_moves(board):

new\_board = make\_move(board, move)

eval = minimax(new\_board, depth-1, False)

max\_eval = max(max\_eval, eval)

return max\_eval

else: min\_eval = math.inf

for move in possible\_moves(board):

new\_board = make\_move(board, move)

eval = minimax(new\_board, depth-1, True)

min\_eval = min(min\_eval, eval)

return min\_eval

**EXP NO. 7 UNIFICATION AND RESOLUTION FOR REAL WORLD PROBLEMS**

AIM:

ALGORITHM:

**PROGRAM:**

from sympy.abc import x

from sympy.logic.boolalg import to\_cnf, Not, And, Or

sentence1 = ForAll(x, Implies(Human(x), Mortal(x)))

sentence2 = Human('Socrates')

unifier = unify(sentence1, sentence2)

negated\_conclusion = Not(Mortal('Socrates'))

resolved = resolve(sentence1.subs(unifier), negated\_conclusion)

if resolved == Or():

print("Socrates is mortal")

else: print("Could not infer that Socrates is mortal")

**EXP NO. 8 IMPLEMENTATION OF KNOWLEDGE REPRESENTATION OF SCHEMES - USE CASES**

AIM:

ALGORITHM:

**PROGRAM:**

import networkx as nx

zoo = nx.DiGraph()

zoo.add\_node('lion')

zoo.add\_node('zebra')

zoo.add\_node('giraffe')

zoo.add\_node('monkey')

zoo.add\_edge('lion', 'zebra', relation='predator')

zoo.add\_edge('lion', 'giraffe', relation='prey')

zoo.add\_edge('zebra', 'giraffe', relation='competes')

zoo.add\_edge('monkey', 'lion', relation='avoids')

print(zoo.edges(data=True))

**EXP NO. 9 IMPLEMENTATION OF UNCERTAIN METHODS**

AIM:

ALGORITHM:

**PROGRAM:**

import numpy as np

from scipy.stats import norm

data = [10, 12, 11, 15, 14, 16, 18, 20, 19, 22]

mu = np.mean(data)

sigma = np.std(data)

model = norm(mu, sigma)

parameters = model.fit(data)

future\_prices = model.rvs(size=1000)

scenarios = []

for i in range(1000):

future\_data = model.rvs(size=len(data))

scenarios.append(future\_data)

threshold = 25

num\_above\_threshold = sum([1 for s in scenarios if max(s) > threshold])

prob\_above\_threshold = num\_above\_threshold / len(scenarios)

print(f"Probability of stock prices reaching {threshold}: {prob\_above\_threshold}")