Experiment - 8
Date - 30/03/2023

Rule Based Inference Problem.

AIM: Developing an optimized technique using an appropriate artificial intelligence algorithm to detect the animal

Algorithm:

- Implication: It is one the logical connectivities which can be represented as $P \rightarrow Q$. It is a Boolean expression.
- Converse: The converse of implication, which means the right hand side proposition goes to the left hand side and vice versa. It can be written as $Q \rightarrow P$
- Contrapositive: The negative of converse is termed as contrapositive, and it can be represented as $Q \rightarrow \neg P$
- Inverse - The negation of implication is called inverse. It can be represented as $\neg P \rightarrow \neg Q$

Code:

```
import sys
def definiteNoun(s):
    s = s.lower().strip()
    if s in ['a', 'e', 'i', 'o', 'u', 'y']:
        return 'an ' + s
    else: return 'a ' + s

def removeArticle(s):
    s = s.lowerStrip()
    if s[0:3] == 'an ': s = s[3:]
    if s[0:2] == 'a ': s = s[2:]
    return s

def makeQuestion(question, yes, no):
    return [question, yes, no]

def isQuestion(p):
    return type(p).__name__ == 'LIST'

def askQuestion(question):
    print('\n%s' % question)
    return sys.stdin.readline().strip().lower()

def getAnswer(question):
    if isQuestion(question):
        return askQuestion(question[0])
    else:
        return askQuestion("Were you thinking about %s?"
                           % definiteNoun(question))
```

```
def gameOver(message):
    global tries
    print("")
    print("\r%s" %message)

def playAgain():
    return answeredYes(askQuestion("Do you want to play?"))

def correctGuess(message):
    global tries
    gameOver(message)

    if playAgain():
        print("")
        tries=0
        return 0

def replaceAnswer(tree, find, replace):
    if not isQuestionTree(tree):
        if tree == find:
            return replace
        else:
            return tree

    else:
        return makeQuestionTree(tree[0],
            replaceAnswer(tree[1], find, replace),
            replaceAnswer(tree[2], find, replace))

def addNewQuestion(wrongAnimal, newques, correct):
    global q
    qv = makeQuestion(newques, correct, wrongAnimal)
    q = replaceAnswer(q, wrongAnimal, qv)
    return q.
    tries=0
    q=(makeQuestion("Does A cat Grass?", 'Cow', 'Tiger')
qv=addNewQuestion('Tiger', 'Does it drink Spirits?', 'Leopard')
qv=addNewQuestion('Leopard', 'Is it fastest animal?', a
qv=0
Print("Imagine an animal I will try to guess
    which one")

try:
    while True:
        ans = answeredYes(getAnswer(qv))
        qv = nextQuestion(qv, ans)
```

```
except KeyboardInterrupt:
    sys.exit(0)
except Exception:
    sys.exit(1)

def nextQuestion(question answer):
    global tries
    tries += 1
    if isQuestion(question):
        if answer:
            return question[1]
        else:
            return question[2]

    else:
        if answer:
            return correctGuess("I knew it!")
        else:
            return MakeNewQuestion(question)
```

Sample Input/Output:

I would like to imagine an animal. I will try to guess which one
you can answer Yes or No.
does it eat grass?
Yes
were you thinking about a cow?
No
I want to learn for the next time. What did you think about?
Deer
Enter the question that would distinguish a deer from a cow:
Does it have horns?
If it asked you this question & you thought about a deer, what would the correct answer be according to you?
Yes
Does it eat grass?
Yes
Does it have horns?
Yes
were you thinking about a deer?
Yes.
I guessed it correct!
& used 3 questions!

Result: Animal Description problem successfully implemented using