**Question:**

*WAP in any high level language to implement the animal game.*

**Background:**

The animal game is a guessing game where the computer ask us a series of questions to try and identify the animal we are thinking about. The program learns as it runs taking input from the user to improve its knowledge base if the answer it had provided was wrong.

The game can be implemented as a Binary tree that searches for the expected answer in the tree’s leaves. The leaves are the answers whereas the non-leaf nodes are the questions. Whenever a question is to be added the node(leaf) where the program ended up is replaced by the sub-tree whose root is the new question and the children are the ‘yes’ answers and the ‘no’ answers. The program also tracks the number of tries made by the computer to reach the answer(leaf). This is however not an effective strategy as the size required by the data structure (tree) would be very large for real life scenarios. The size of the data structure being large will also cause the searching time to be high which might to be sufficient for real life problems.

**Methodology:**

The program was written in python and the list data structure was used to implement the binary tree. Each node is a list having three elements:

1. The actual question.
2. The right answer of question or another list of three elements that signifies the question to be asked if the user responded as ‘yes’ for the earlier question.
3. The Wrong answer of the question or another list of three elements that signifies the question to be asked if the user responded as ‘no for the earlier question.

For example

**['Can it swim ?', 'Whale', 'Pigeon']:**

This list has a question ‘Can it swim?’, If the user inputs ‘yes’ whale is the answer else Pigeon is the answer.

**['Can it swim ?', 'Whale', ['Does it have a tail?', 'Cat', 'Pigeon']]**

Here, if the answer is ‘no’ then instead of giving pigeon as the answer another question ‘Does it have a tail?’ is asked and so on.

**Program:**

import sys

def toBool(answer):

if answer[0]=='y':

return True

else:

return False

def makeQuestion(question, yes, no):

return [question, yes, no]

def createQuestion(question, rightAns, wrongAns):

return [question, rightAns, wrongAns]

def validate(ans):

if(ans=="yes" or ans=="no"):

return True

else:

return False

def askQuestion(question):

if (type(question).\_\_name\_\_ == "list"):

return (raw\_input(question[0])).lower()

else:

return raw\_input("Were you thinking about %s?" % question).lower()

def playAgain():

return toBool(raw\_input("Do you want to play again?"))

def nextQuestion(question, answer):

global tries

tries += 1

if (type(question).\_\_name\_\_ == "list"):

if answer:

return question[1]

else:

return question[2]

else:

if answer:

print("I told you I can do it. I tool " + str(tries) + " tries" )

tries = 0

if playAgain():

tries = 0

return firstQues

else:

sys.exit(0)

else:

return makeNewQuestion(question)

def replaceAnswer(tree, find, replace):

if not (type(tree).\_\_name\_\_ == "list"):

if tree == find:

return replace

else:

return tree

else:

return makeQuestion(tree[0], replaceAnswer(tree[1], find, replace), replaceAnswer(tree[2], find, replace))

def makeNewQuestion(wrongAns):

global firstQues, tries

correctAns = raw\_input("I give up :(, What were you thinking about ??? ")

question = raw\_input("Enter a question that distinguishes %s from %s. "%(correctAns, wrongAns))

answer = toBool(raw\_input("If I asked you this question and you thought about %s, what would the correct answer be?" % correctAns).lower())

if answer:

newQuestion = [question, correctAns, wrongAns]

else:

newQuestion = [question, wrongAns, correctAns]

ques = replaceAnswer(firstQues, wrongAns, newQuestion)

firstQues = ques

# print(str(answer) , ques)

tries = 0

return ques

tries = 0

firstQues = createQuestion("Can it swim ?", "Whale", "Pegion")

ques = firstQues

# print(ques)

while True:

ans = askQuestion(ques)

while not validate(ans):

print("answer in only yes or no")

ans = askQuestion(ques)

ques = nextQuestion(ques, toBool(ans))

**Output:**

