**Kriston Pal Lab 1 Report**

**0511 Date: 15/02/2018**

**Question:**

Write a Program in any high-level language to implement the animal guessing game.

**Background:**

We are given a problem that is to be solved using tree especially Binary tree, a data structure where a node has at most two children, usually referred to as the “left" child and the “right" child. To illustrate binary tree, we solved animal guessing problem in which computer asks many questions and based on our answer it guesses the name of the animal.

In the constructed binary tree, leaves are the answers and 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. Here in this problem, solution also tracks for the number of steps(questions) that have been asked to guess the name of animals.

**Implementation:**

I have used Python as a high-level language to solve this animal guessing problem where implementation of Binary tree is illustrated. I chose Python as it is easy to implement AI concept somehow so that device can be trained to increase intelligence by adding knowledge to it.

In this game, initially a question is asked, and user need to answer in yes or no. Based on the answers given by user, computer guesses the name of that animal. If it fails to guess then, it allows user to add animal name and distinguishing questions as well.

**For example:**

[‘Can it fly?', **"Bat"**, **"Elephant"**]:

Here the question is Can it fly? If user enters yes, then the correct answer is Bat else if no then, Elephant may be correct answer. If elephant is also not correct answer, then it prints message as I give Up and allows you to train it by adding animal’s names and questions.

A decision tree for this game is;

A screenshot of a social media post

Description generated with very high confidence

**Program (Source Code):**

*#AI\_LAB1  
#Kriston Pal  
#0511***import** sys  
  
**from** pip.\_vendor.colorama **import** Fore, Back, Style *#import to use color for text  
  
# print(Fore.RED + 'some red text')*name = str(input(**"What is your name? "**))  
  
print (**"Hello @"**,Fore.BLUE +name, Fore.RED +**"\n WELCOME TO THE CHALLENGE "**, (Style.RESET\_ALL)+ Fore.GREEN +  
 **"\n I can guess the name of animal that you have in your Mind"**)  
  
print(Fore.LIGHTYELLOW\_EX+**'You can make me more intelligent by training me'**) *#created simple user interface*print(Style.RESET\_ALL) *#resets color use***def** Boolean(answer): *#answer can be either true or false* **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** test(ans): *#validate the answer given by player should be either yes or no* **if**(ans==**"yes" or** ans==**"no"**):  
 **return True  
 else**:  
 **return False  
  
def** askQuestion(question): *#checks whether there is question in list left or not?* **if**(type(question).\_\_name\_\_==**"list"**):  
 **return** str(input(question[0])).lower()  
 **else**:  
 **return** str(input(**"Were you thinking about %s?"** % question).lower())  
  
  
**def** playAgain(): *#whether user want to plain again or quit game* **return** Boolean(str(input(**"Do you want to play again?"**)))  
  
**def** nextQuestion(question, answer): *#returns questions from the list and prints acknowldgement* **global** count  
 count += 1  
 **if** (type(question).\_\_name\_\_ == **"list"**):  
 **if** answer:  
 **return** question[1]  
 **else**:  
 **return** question[2]  
 **else**:  
 **if** answer:  
 print(Fore.MAGENTA+**"Yes I did it. I tried for "** + str(count) + **" times"**)  
 print(Style.RESET\_ALL)  
 count = 0  
 **if** playAgain(): *#starts game again* count = 0  
 **return** firstQues  
 **else**:  
 sys.exit(0) *#exits game* **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, count  
 correctAns =str(input(**"I give up :(, What were you thinking about ??? \t"**))  
 question = str(input(**"Enter a question that distinguishes %s from %s. "**%(correctAns, wrongAns)))  
 answer = Boolean(str(input(**"If I asked you this question and you thought about %s, what would the correct answer be?\t"** % correctAns).lower()))  
 **if** answer:  
 newQuestion = [question, correctAns, wrongAns]  
 **else**:  
 newQuestion = [question, wrongAns, correctAns]  
 ques = replaceAnswer(firstQues, wrongAns, newQuestion) *#after training game starts from initial state* firstQues = ques  
 count = 0  
 **return** ques  
  
count = 0 *#initially count is zero*firstQues = createQuestion(**"Can it fly ?\t"**, **"Bat"**, **"Elephant"**)

*#call createQuestion function where (question, rightAns, wrongAns) are the parameters*ques = firstQues

*# print(ques)***while True**: *#if answer is True* ans = askQuestion(ques)  
 **while not** test(ans): *#if the user reply is not yes or no* print(**"Your answer must be either yes or no\n "**)  
 ans = askQuestion(ques)  
 ques = nextQuestion(ques, Boolean(ans))

**Output:**

A screenshot of a cell phone

Description generated with very high confidence