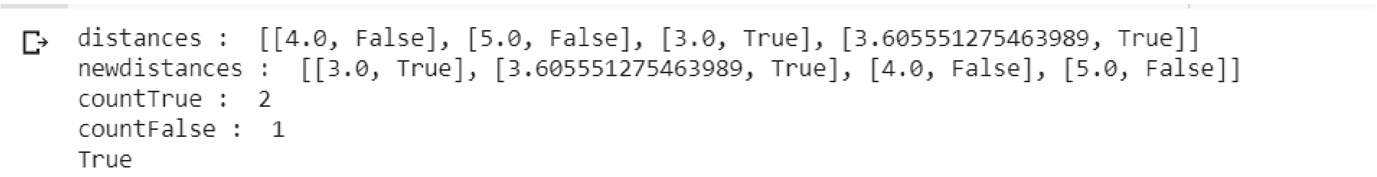
Ritish Gupta 101983057 COE-25

AI Lab Evaluation 2

Q1. Perform kNN- classification algorithm on the following dataset predict the class for x(P1=3 and P2=7), k=3.

```
import math

# given data

dataset = {
  "P1" : [7, 7, 3, 1],
  "P2" : [7, 4 ,4 ,4],
  "Class" : [False, False, True, True]
}
k = 3
testcase = [3,7]


# sort the list of [distance,Class]

def sort_distances(newdistances):
  l = len(newdistances)
  for i in range(l):
    for j in range(l):
      if newdistances[i][0] < newdistances[j][0]:
        swap = newdistances[i]
        newdistances[i] = newdistances[j]
        newdistances[j] = swap
  return newdistances



distances = []
nearest_neighbors = []

def knn(dataset, testcase, k):
  # claculate euclidean distance
  for i in range(len(dataset["P1"])):
    a = abs(dataset["P1"][i] - testcase[0])**2 + abs(dataset["P2"][i]-
testcase[1])**2
    b = [math.sqrt(a),dataset["Class"][i]]
    distances.append(b)
  print("distances : ", distances)
  # sort the distances list and put in newdistances
  newdistances = sort_distances(distances)
  print("newdistances : ", newdistances)

  # pick top k distances
  countTrue = 0
  countFalse = 0
  for i in range(k):
    nearest_neighbors.append(newdistances[i])
    if newdistances[i][1] == True:
      countTrue+= 1
    else:
      countFalse+= 1

  # predict the answer
  print("countTrue : " ,countTrue, "\ncountFalse : " , countFalse)
  if countTrue > countFalse:
    return "True"
  else:
    return "False"



answer = knn(dataset, testcase, k)
print(answer)
```

OUTPUT:-

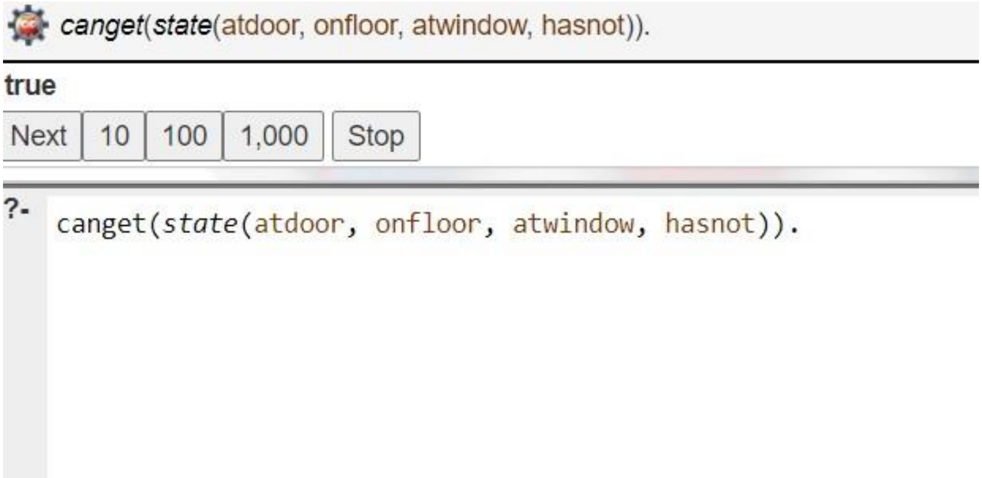


```
distances :  [[4.0, False], [5.0, False], [3.0, True], [3.605551275463989, True]]
newdistances :  [[3.0, True], [3.605551275463989, True], [4.0, False], [5.0, False]]
countTrue :  2
countFalse :  1
True
```

Q2. Solve the monkey-banana problem using prolog, with the scenario: Monkey is on the floor, at the door. A block is on the floor, at the window. Banana is hanging from the roof in the middle of the room. Problem is "How the monkey can get the banana". The monkey can perform the following actions: Walk on the floor, climb the box, push the box around (if it is besides the box), grasp the banana if it is standing on the box directly under the banana

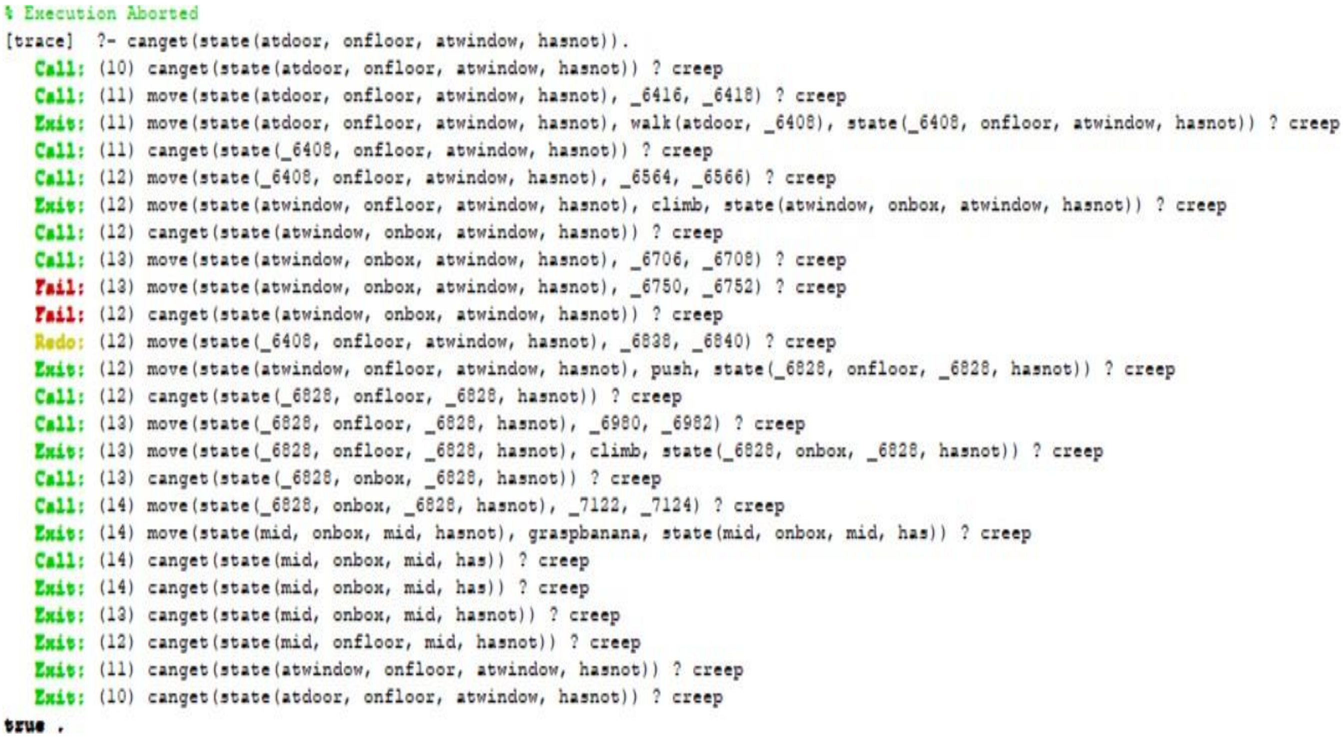
```
move(state(mid, onbox, mid, hasnot), graspbanana, state(mid, onbox,
mid, has)).
move(state(POS, onfloor, POS, H), climb, state(POS, onbox, POS, H)).
move(state(POS1, onfloor, POS1, H), push, state(POS2, onfloor, POS2,
H)).
move(state(POS1, onfloor, B, H), walk(POS1,POS2), state(POS2, onfloor,
B, H)).
canget(state(_,_,_,has)).
canget(State1):-
    move(State1,_,State2),
    canget(State2).
```

OUTPUT:-



```
canget(state(atdoor, onfloor, atwindow, hasnot)).

true
```

```
?- canget(state(atdoor, onfloor, atwindow, hasnot)).
```



```
% Execution Aborted
[trace]  ?- canget(state(atdoor, onfloor, atwindow, hasnot)).
   Call: (10) canget(state(atdoor, onfloor, atwindow, hasnot)) ? creep
   Call: (11) move(state(atdoor, onfloor, atwindow, hasnot), _6416, _6418) ? creep
   Exit: (11) move(state(atdoor, onfloor, atwindow, hasnot), walk(atdoor, _6408), state(_6408, onfloor, atwindow, hasnot)) ? creep
   Call: (11) canget(state(_6408, onfloor, atwindow, hasnot)) ? creep
   Call: (12) move(state(_6408, onfloor, atwindow, hasnot), _6564, _6566) ? creep
   Exit: (12) move(state(atwindow, onfloor, atwindow, hasnot), climb, state(atwindow, onbox, atwindow, hasnot)) ? creep
   Call: (12) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep
   Call: (13) move(state(atwindow, onbox, atwindow, hasnot), _6706, _6708) ? creep
   Fail: (13) move(state(atwindow, onbox, atwindow, hasnot), _6750, _6752) ? creep
   Fail: (12) canget(state(atwindow, onbox, atwindow, hasnot)) ? creep
   Redo: (12) move(state(_6408, onfloor, atwindow, hasnot), _6838, _6840) ? creep
   Exit: (12) move(state(atwindow, onfloor, atwindow, hasnot), push, state(_6828, onfloor, _6828, hasnot)) ? creep
   Call: (12) canget(state(_6828, onfloor, _6828, hasnot)) ? creep
   Call: (13) move(state(_6828, onfloor, _6828, hasnot), _6980, _6982) ? creep
   Exit: (13) move(state(_6828, onfloor, _6828, hasnot), climb, state(_6828, onbox, _6828, hasnot)) ? creep
   Call: (13) canget(state(_6828, onbox, _6828, hasnot)) ? creep
   Call: (14) move(state(_6828, onbox, _6828, hasnot), _7122, _7124) ? creep
   Exit: (14) move(state(mid, onbox, mid, hasnot), graspbanana, state(mid, onbox, mid, has)) ? creep
   Call: (14) canget(state(mid, onbox, mid, has)) ? creep
   Exit: (14) canget(state(mid, onbox, mid, has)) ? creep
   Exit: (13) canget(state(mid, onbox, mid, hasnot)) ? creep
   Exit: (12) canget(state(mid, onfloor, mid, hasnot)) ? creep
   Exit: (11) canget(state(atwindow, onfloor, atwindow, hasnot)) ? creep
   Exit: (10) canget(state(atdoor, onfloor, atwindow, hasnot)) ? creep
true .
```