Ahsanullah University of Science and Technology Department of Computer Science and Engineering

Course No: CSE4108

Course Title: Artificial Intelligence Lab

Assignment No: 01

Submitted by

Name: Rizeya Rabbi Reyad

ID: 160104082

Lab Group: B2

Question 3: Write code in python and prolog to find grandparents of somebody.
Answer:

Python code

```
tupleList1=[('parent', 'Hasib', 'Rakib'),('parent', 'Rakib', 'Sohel'),
('parent', 'Rakib', 'Rebeka'),('parent', 'Rashid', 'Hasib')]

X=str(input("Grandchild:"))
print('Grandparent:', end=' ')
i=0 while(i<=3):
   if ((tupleList1[i][0] == 'parent')&( tupleList1[i][1] == X)):
      for j in range(4):            if ((tupleList1[j][0] == 'parent') & (
tupleList1[i][2] == tupleList1[j][1])):
            print(tupleList1[j][2], end=' ')
i=i+1
```

Prolog code

```
parent('Hasib' , 'Rakib'). parent('Rakib' ,
'Sohel'). parent('Rakib' , 'Rebeka'). parent('Rashid' ,
'Hasib').

grandparant(X, Z) :- parent(X, Y), parent(Y, Z).

findGp :- write('Grandchild: '), read(X), write('Grandparent: '),
grandparant(X, Gc), write(Gc), tab(5), fail.
findGp.
```

Question 4: Enrich the KB demonstrated above with 'brother', 'sister', 'uncle' and 'aunt' rules in Python and Prolog.
Answer:

Python code

```
tupleList1 = [('parent', 'Hasib', 'Rakib'),('parent', 'Rakib', 'Sohel'), ('parent', 'Hafsa',
'Rakib'),('parent', 'Rafa', 'Sohel'), ('parent', 'Rakib', 'Rebeka'),('parent', 'Rashid', 'Hasib')]
male = ['Hasib', 'Sohel', 'Rashid'] female = ['Rebeka', 'Hafsa', 'Rafa']

def optionPrint():
  print("Choose any option:")
print("1. Find Grandparent")
print("2. Find Brother")
print("3. Find Sister")    print("4.
Find Uncle")    print("5. Find
Aunt")

def findGP(X = "empty"):
if X == "empty":
      X = str(input("Grandchild:"))    print('Grandparent:', end=' ')    i = 0
while (i < 6):       if ((tupleList1[i][0] == 'parent') & (tupleList1[i][1] == X)):
for j in range(6):             if ((tupleList1[j][0] == 'parent') & (tupleList1[i][2]
== tupleList1[j][1])):
                print(tupleList1[j][2], end=' ')
i = i + 1

def findBrother(X = "empty"):
if X == "empty":
      X = str(input("Your name:"))
print('Brother:', end=' ')     i = 0
while (i < 6):
      if ((tupleList1[i][0] == 'parent') & (tupleList1[i][1] == X)):
         for j in range(6):
            if ((tupleList1[j][0] == 'parent') & (tupleList1[i][2] == tupleList1[j][2])):
               if tupleList1[j][1] in male:
                  print(tupleList1[j][1], end=' ')
      i = i + 1

def findSister(X = "empty"):
  if X == "empty":
      X = str(input("Your name:"))
print('Sister:', end=' ')
  i = 0
  while (i < 6):
      if ((tupleList1[i][0] == 'parent') & (tupleList1[i][1] == X)):
         for j in range(6):             if ((tupleList1[j][0] == 'parent') &
(tupleList1[i][2] == tupleList1[j][2])):             if tupleList1[j][1] in female:
                  print(tupleList1[j][1], end=' ')
      i = i + 1

def findUncle(X = "empty"):
if X == "empty":
    X = str(input("Your name:"))    print('Uncle:', end=' ')
i = 0   while (i < 6):       if ((tupleList1[i][0] == 'parent') &
(tupleList1[i][1] == X)):
         findBrother(tupleList1[i][2])
i = i + 1

def findAunt(X = "empty"):
if X == "empty":
    X = str(input("Your name:"))
print('Aunt:', end=' ')    i = 0
while (i < 6):
    if ((tupleList1[i][0] == 'parent') & (tupleList1[i][1] == X)):
         findSister(tupleList1[i][2])
i = i + 1


optionPrint() x = int(input("Enter
your choice: "))

if x == 1:
findGP() elif x
== 2:
findBrother()
elif x == 3:
findSister()
elif x == 4:
findUncle()
elif x == 5:
findSister()
```

Prolog code parent('Hasib', 'Rakib'). parent('Hafsa', 'Rakib').
parent('Rakib', 'Sohel'). parent('Rafa', 'Sohel'). parent('Rakib',
'Rebeka'). parent('Rashid', 'Hasib').

```
male('Hasib'). male('Sohel'). male('Rashid'). female('Rebeka').
female('Hafsa'). female('Rafa').

grandparant(X, Z) :- parent(X, Y), parent(Y, Z).
brother(X, Y) :- parent(X, Z), parent(Y, Z), male(Y).
```

```
sister(X, Y) :- parent(X, Z), parent(Y, Z), female(Y).
uncle(X, Z) :- parent(X, Y), brother(Y, Z). aunt(X,
Y) :- parent(X, Z), sister(Z, Y).

findBrother :- write('Your name: '), read(X), write('Your brother: '), brother(X, Br), write(Br), tab(5),
fail. findBrother.
findSister :- write('Your name: '), read(X), write('Your sister: '), sister(X, S), write(S), tab(5), fail.
findSister.
findUncle :- write('Your name: '), read(X), write('Your uncle: '), uncle(X, Un), write(Un), tab(5), fail.
findUncle.
findAunt :- write('Your name: '), read(X), write('Your aunt: '), aunt(X, An), write(An), tab(5), fail.
findUncle.
```