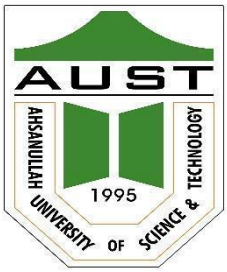


Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE4108**
Course Title : **Artificial Intelligence Lab**
Assignment No : **01**

Date of Performance : **27.11.2022**
Date of Submission : **10.12.2022**

Submitted To : **Mr. Faisal Muhammad Shah & Mr. Raihan Tanvir**

Submitted By-
Group : **C_2**
Name : **Ashiqul Islam**
Id : **190104140**
Section : **C**

Prolog Code:

```
parent('Hasib' , 'Rakib').
parent('Rakib' , 'Sohel').
parent('Rakib' , 'Rebeka').
parent('Rashid' , 'Hasib').
parent('Rakib', 'Asif').
parent('Rakib', 'Saber').
parent('Rabeya', 'Asif').
parent('Rabeya', 'Sohel').
parent('Hasib', 'Rakhi').
parent('Samir', 'Rabeya').
parent('Samir', 'Akib').
parent('Samir', 'Tanny').

male('Hasib').
male('Rakib').
male('Sohel').
male('Rashid').
male('Ashiq').
male('Asif').
male('Samir').
male('Akib').
male('Saber').

female('Rebeka').
female('Rabeya').
female('Rakhi').
female('Tanny').
```

```
brother(X, Z) :- parent(P, X), parent(P, Z), male(Z), not(X = Z).

findBrother :- write('Name: '), read(X), nl, write('Brother/s of
'), print(X), write(': '), nl,
   brother(X, Z), write(Z), tab(4), fail.
findBrother.


sister(X, Z) :-  parent(P, X), parent(P, Z), female(Z), not(X = Z).

findSister :- write('Name: '), read(X), nl, write('Sister/s of '),
print(X), write(': '), nl,
   sister(X, Z), write(Z), tab(4), fail.
findSister.


uncle(X, Z) :- parent(P, X), parent(G, P), parent(G, Z), male(Z),
not(Z = P).

findUncle :- write('Name: '), read(X), nl, write('Uncle/s of '),
print(X), write(': '), nl,
   uncle(X, Z), write(Z), tab(4), fail.
findUncle.


aunt(X, Z) :- parent(P, X), parent(G, P), parent(G, Z),
female(Z), not(Z = P).

findAunt :- write('Name: '), read(X), nl, write('Aunt/s of '),
print(X), write(': '), nl,
   aunt(X, Z), write(Z), tab(4), fail.
findAunt.
```

Explanation: At first we've declared the facts in out Knowledgebase. For example, in parent('Hasib' , 'Rakib') Hasib is the father of Rakib; male('Hasib') means Hasib is a male and female('Rebeka') means Rebeka is a female in our Knowledgebase

brother(X, Z): Here X is input whose Brother we will find. If X is brother of Z then their parent must be same (that is Y) and Z must be male.

sister(X, Z) :- Here X is input whose Sister we will find. If X is sister of Z then their parent must be same (that is Y) and Z must be female.

uncle(X, Z) :- Here X is input whose Uncle we will find. We first find the parent of X that is P. and we find the parent of P that is (G), the grandparent of X. Then we find the children of G that is Z. We check so that the value of Z (that is the uncle) is not the same as P (because parent of a person can not be his uncle) and Z must be male.

Aunt(X, Z) :- Here X is input whose Aunt we will find. We first find the parent of X that is P. and we find the parent of P that is (G), the grandparent of X. Then we find the children of G that is Z. We check so that the value of Z (that is the aunt) is not the same as P (because parent of a person can not be his aunt) and Z must be female.

Screenshots of output:

assignment1.pl

```
parent('Hasib' , 'Rakib').
parent('Rakib' , 'Sohel').
parent('Rakib' , 'Rebeka').
parent('Rashid' , 'Hasib').
parent('Rakib', 'Asif').
parent('Rakib', 'Saber').
parent('Rabeya', 'Asif').
parent('Rabeya', 'Sohel').
parent('Hasib', 'Rakhi').
parent('Samir', 'Rabeya').
parent('Samir', 'Akib').
parent('Samir', 'Tanny').

male('Hasib').
male('Rakib').
male('Sohel').
male('Rashid').
male('Ashiq').
male('Asif').
male('Samir').
male('Akib').
male('Saber').

female('Rebeka').
female('Rabeya').
female('Rakhi').
female('Tanny').

brother(X, Z) :- parent(P, X), parent(P, Z), male(Z), not(X = Z).
findBrother :- write('Name: '), read(X), nl, write('Brother/s of '), print(X), write(': '), nl,
    brother(X, Z), write(Z), tab(4), fail.

sister(X, Z) :-  parent(P, X), parent(P, Z), female(Z), not(X = Z).
findSister :- write('Name: '), read(X), nl, write('Sister/s of '), print(X), write(': '), nl,
    sister(X, Z), write(Z), tab(4), fail.

uncle(X, Z) :- parent(P, X), parent(G, P), parent(G, Z), male(Z), not(Z = P).
findUncle :- write('Name: '), read(X), nl, write('Uncle/s of '), print(X), write(': '), nl,
    uncle(X, Z), write(Z), tab(4), fail.

aunt(X, Z) :- parent(P, X), parent(G, P), parent(G, Z), female(Z), not(Z = P).
findAunt :- write('Name: '), read(X), nl, write('Aunt/s of '), print(X), write(': '), nl,
    aunt(X, Z), write(Z), tab(4), fail.
```

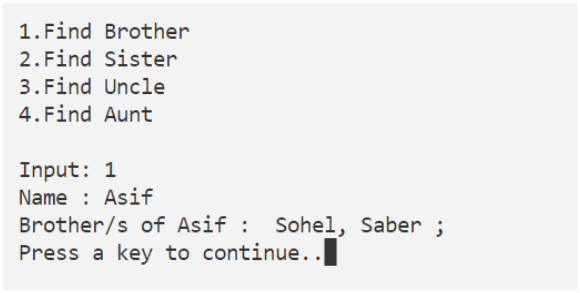
```
% d:/4.1/ai/lab/assignment1 compiled 0.00 sec, -7 clauses
?- findBrother.
Name: 'Asif'.

Brother/s of 'Asif':
Sohel    Saber    Sohel
false.

?- findSister.
Name: 'Asif'.

Sister/s of 'Asif':
Rebeka
false.

?- findUncle.
Name: 'Asif'.

Uncle/s of 'Asif':
Akib
false.

?- findAunt.
Name: 'Asif'.

Aunt/s of 'Asif':
Rakhi    Tanny
false.

?-
```

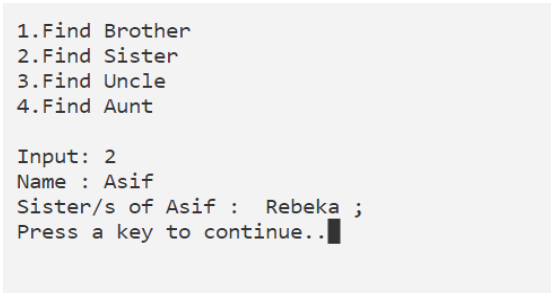
Python Code:

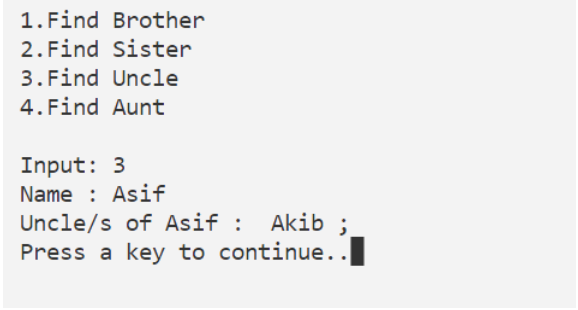
```
import os
ParentKB =  [('parent', 'Hasib', 'Rakib'),
        ('parent', 'Rakib', 'Sohel'),
        ('parent', 'Rakib', 'Rebeka'),
        ('parent', 'Rashid', 'Hasib'),
        ('parent', 'Rakib', 'Asif'),
        ('parent', 'Rakib', 'Saber'),
        ('parent', 'Rabeya', 'Asif'),
        ('parent', 'Rabeya', 'Sohel'),
        ('parent', 'Hasib', 'Rakhi'),
        ('parent', 'Samir', 'Rabeya'),
        ('parent', 'Samir', 'Akib'),
        ('parent', 'Samir', 'Tanny')]

MaleKB = ['Hasib', 'Rakib', 'Sohel', 'Rashid', 'Ashiq', 'Asif', 'Samir', 'Akib',
'Saber']
FemaleKB = ['Rebeka', 'Rabeya', 'Rakhi', 'Tanny']

def findSibling(relation, name):
    parents = []
    siblings = []

    for i in range(len(ParentKB)):
        if (ParentKB[i][2] == name):
            parents.append(ParentKB[i][1])

    for i in range(len(parents)):
        for j in range(len(ParentKB)):
            if(parents[i] == ParentKB[j][1] and ParentKB[j][2] != name ):
                if(relation == "Brother" and ParentKB[j][2] in MaleKB and
ParentKB[j][2] not in siblings ):
                    siblings.append(ParentKB[j][2])
                elif(relation == "Sister" and ParentKB[j][2] in FemaleKB and
ParentKB[j][2] not in siblings):
                    siblings.append(ParentKB[j][2])
    if(len(siblings) != 0):
        print(*siblings , end= " ;", sep = ", ")


def findRelative(relation, name):
    parents = []

    for i in range(len(ParentKB)):
        if (ParentKB[i][2] == name):
            parents.append(ParentKB[i][1])
```

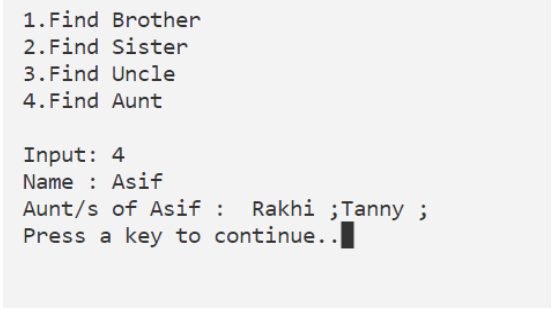
```
for i in range(len(parents)):
        if(relation == "Uncle"):
            findSibling("Brother", parents[i])
        elif(relation == "Aunt"):
            findSibling("Sister", parents[i])

while(True):
    os.system('cls')
    print("1.Find Brother \n2.Find Sister \n3.Find Uncle \n4.Find Aunt \n")
    userInput = int(input("Input: "))
    X = input("Name : ")

    if(userInput == 1):
        print("Brother/s of {0} : ".format(X), end=' ')
        findSibling("Brother", X)
    elif(userInput == 2):
        print("Sister/s of {0} : ".format(X), end=' ')
        findSibling("Sister", X)
    elif(userInput == 3):
        print("Uncle/s of {0} : ".format(X), end=' ')
        findRelative("Uncle", X)
    elif(userInput == 4):
        print("Aunt/s of {0} : ".format(X), end=' ')
        findRelative("Aunt", X)

    input("\nPress a key to continue..")
```

Screenshots of output:









Explanation:

Our knowledge base is made of 3 lists. KB1 is a list of tuples which holds the relation between two persons that are in our knowledgebase. MaleKB is a list of persons in our KB1 who are male. FemaleKB is a list of persons in KB1 who are female.

The *findSibling*() function takes two parameters, one, is the relation(Brother/Sister) that is to be found, and second, the *name* for whom we want the output. We iterate through our KB1 list and find out his parents and store their name in the parents list. Then we iterate through the parents list, and for each element of the parents list we find out their children according to our need (We determine that is we need to find male children or female children by checking the value of the *relation* parameter of this function) and add them to our *siblings* list. We also check if the name of the children is same as the given name. If so, then we do not add it to our *siblings* list. Then at last we print the *siblings* list.

The *findRelative()* function takes two parameters, one, is the relation(Brother/Sister) that is to be found, and second, the *name* for whom we want the output. Then we find out the parents of the given input name and save them to the parents list. Then, we iterate the list and pass each of the element to the *findSibling()* function to print the siblings of the parents that is the Uncle/Aunt of the given *name*.

At the end we use a while loop to make a menu driven program to take input from user and find his Brother/Sister/Uncle/Aunt automatically.