



Ahsanullah University of Science and Technology

Department of Computer Science & Engineering

| | |
|----------------|-----------------------------|
| Course No. | CSE 4108 |
| Course Name | Artificial Intelligence Lab |
| Assignment No. | 01 |

Submitted To:

| | |
|-------------------------|-------------------------|
| Md. Siam Ansary | Tonmoy Hossain |
| Department of CSE, AUST | Department of CSE, AUST |

Submitted By:

| | |
|---------------------|------------------------|
| Name | Tahiya Ahmed Chowdhury |
| ID No. | 17.02.04.048 |
| Session | Fall – 2020 |
| Section | A (A2) |
| Date of Submission: | July 4, 2021 |

Question 03: Modify the Python and Prolog codes demonstrated above to find the grandparents of somebody.

From the provided codes in our lab material we can see four facts, one rule for determining grandchild and also a python code for the same function. Now using the same fact, we are modifying the code to find the grandparent of someone that is basically the reverse of the given function.

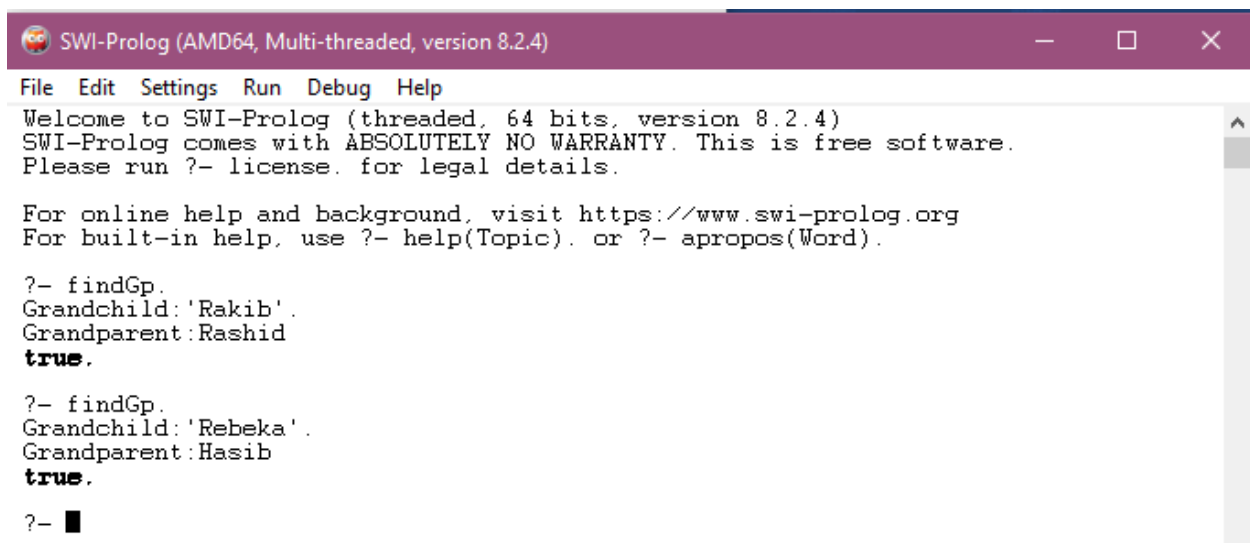
Prolog code:

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

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

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

Prolog output:



```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- findGp.
Grandchild: 'Rakib'.
Grandparent: Rashid
true.

?- findGp.
Grandchild: 'Rebeka'.
Grandparent: Hasib
true.

?- █
```

Analyzing code and above input and output:

For determining the grandparent in prolog we applied the logic from the given facts that if Rakib is Soheli's father and Hasib is Rakib's father then Hasib is Soheli's grandparent. This rule applies for everyone. In the input/output section we may see that whenever we are entering grandchild's name, the grandparent's name is extracted.

Python code:

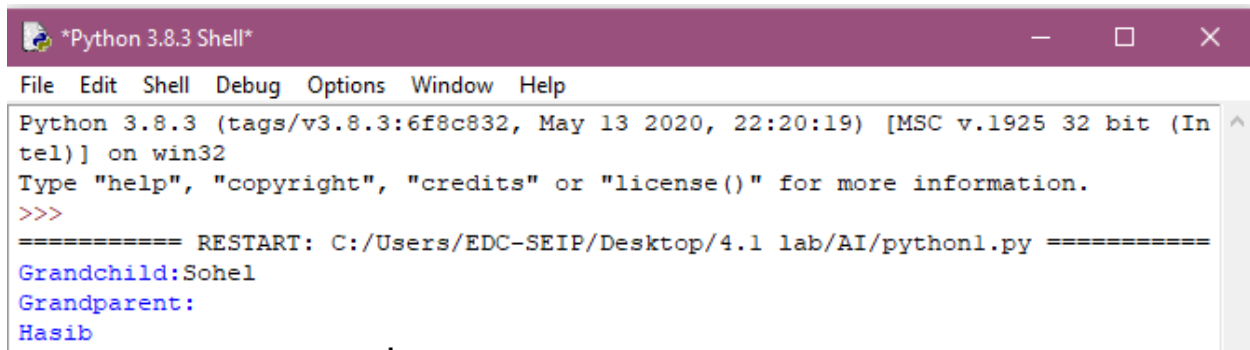
```
tuplelist1 = [ ('parent','Hasib','Rakib') , ('parent','Rakib','Sohel') , ('parent','Rakib','Rebeka'),  
( 'parent','Rashid','Hasib') , ('parent','Sohel','Ann') , ('parent','Rebeka','Sam') ]
```

```
X=str(input("Grandchild:"))  
print('Grandparent:')
```

```
i,j=0,0
```

```
while(i<=3):  
    if((tuplelist1[i][0] == 'parent')&(tuplelist1[i][2] == X)):  
        for j in range(4):  
            if( (tuplelist1[j][0] == 'parent') & (tuplelist1[i][1] == tuplelist1[j][2]) ):  
                print(tuplelist1[j][1]," ")  
  
        i = i + 1
```

Python output:



```
*Python 3.8.3 Shell*  
File Edit Shell Debug Options Window Help  
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/EDC-SEIP/Desktop/4.1 lab/AI/python1.py =====  
Grandchild:Sohel  
Grandparent:  
Hasib
```

Analyzing code and above input and output:

For the python code, we converted the facts into tuples and took grandchild's name as input. The outer loop searches the grandchild's name in the tuple and once it is found, it is passed to the inner loop to check whether the grandchild's parent matches with any child's parent. If so then the name of that parent who also happens to be the grandfather of the given entry is returned.

Question 04: Enrich the KB demonstrated above with 'brother', 'sister', 'uncle' and 'aunt' rules in Python and Prolog.

For enriching the provided KB with 'brother', 'sister', 'uncle', 'aunt' let us first add two more facts : ('parent','Sohel','Ann'),
('parent','Rebeka','Sam') because the given facts were not adequate for determining the relations such as uncle and aunt.

Prolog code:

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

```
male('Hasib').
male('Rakib').
male('Sohel').
male('Rashid').
```

```
female('Rebeka').
```

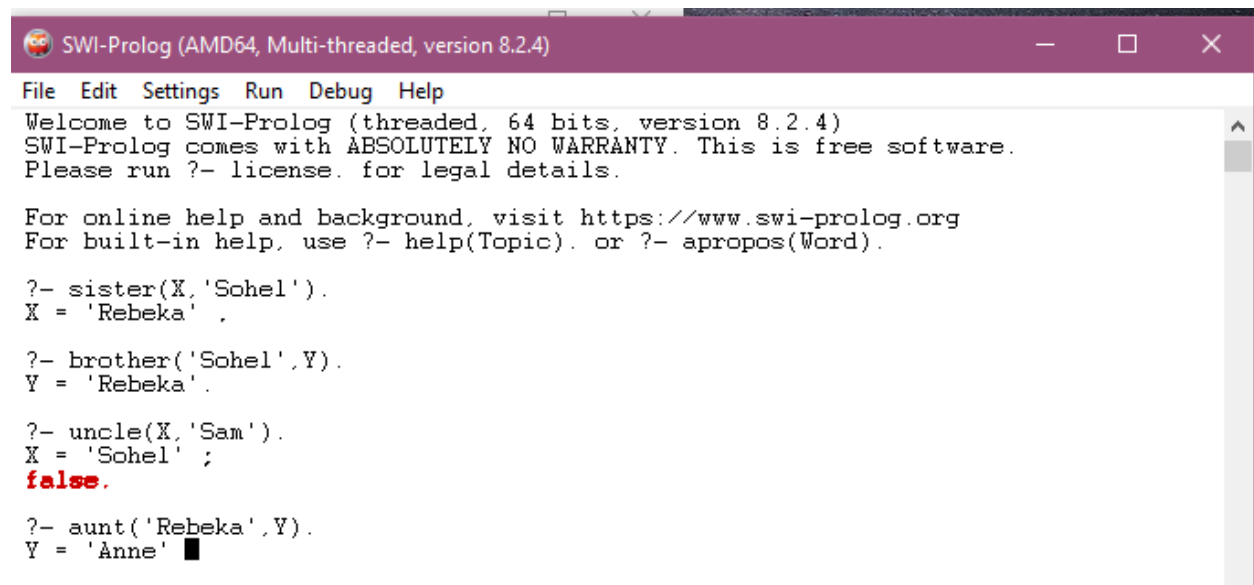
```
sister(X,Y):-
    parent(Z,X),parent(Z,Y),female(X),X\=Y.
```

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

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

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

Prolog output:



```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- sister(X,'Sohel').
X = 'Rebeka' ;

?- brother('Sohel',Y).
Y = 'Rebeka' ;

?- uncle(X,'Sam').
X = 'Sohel' ;
false.

?- aunt('Rebeka',Y).
Y = 'Anne' ;
```

Analyzing code and above input and output:

While determining the siblings we followed a simple logic : if two children have same parent then they are sibling. Then to determine whether they are brothers or sisters, the new facts to distinguish male and female was introduced. The the logic got down to if two entries are siblings and the first one is a male then it falls under brother category and if it is female then it falls under the sister category. After solving the brother and sister rule, we introduced the uncle and aunt rule : if Soheli and Rebeka are siblings then they are uncles and aunts to each others children.

Python code:

```
tuplelist1 = [ ('parent','Hasib','Rakib') , ('parent','Rakib','Sohel') , ('parent','Rakib','Rebeka') ,  
('parent','Rashid','Hasib') , ('parent','Sohel','Ann') , ('parent','Rebeka','Sam') ]
```

```
tuplelist2 = [ ('male','Hasib') , ('male','Sohel') , ('male','Rakib') , ('male','Rashid') , ('male','Sam') ,  
('female','Ann') , ('female','Rebeka') ]
```

```
X1=str(input("Sibling name for finding brother:"))  
print('Brother:')  
i1,j1,k1=0,0,0  
while(i1<=5):  
    if((tuplelist1[i1][0] == 'parent')&(tuplelist1[i1][2] == X1)):  
        for j1 in range(6):  
            if( (tuplelist1[j1][0] == 'parent') & ((tuplelist1[i1][1] == tuplelist1[j1][1]) &  
(tuplelist1[j1][2]!=X1))):  
                for k1 in range(7):  
                    if((tuplelist2[k1][1] == tuplelist1[j1][2]) & (tuplelist2[k1][0] == 'male')):  
                        print(tuplelist1[j1][2]," ")  
  
        i1 = i1 + 1
```

```
X2=str(input("Sibling name for finding sister:"))  
print('Sister:')  
  
i2,j2,k2=0,0,0  
while(i2<=5):  
    if((tuplelist1[i2][0] == 'parent')&(tuplelist1[i2][2] == X2)):  
        for j2 in range(6):  
            if( (tuplelist1[j2][0] == 'parent') & ((tuplelist1[i2][1] == tuplelist1[j2][1]) &  
(tuplelist1[j2][2]!=X2))):  
                for k2 in range(7):  
                    if((tuplelist2[k2][1] == tuplelist1[j2][2]) & (tuplelist2[k2][0] == 'female')):  
                        print(tuplelist1[j2][2]," ")
```

```

i2 = i2 + 1
X3=str(input("Neice/Nephew name for finding uncle:"))
print('Uncle:')
m,n=0,0

while(m<=5):
    if((tuplelist1[m][0] == 'parent')&(tuplelist1[m][2] == X3)):
        for n in range(6):
            if( (tuplelist1[n][0] == 'parent') & (tuplelist1[m][1] == tuplelist1[n][2]) ):
                i3,j3,k3=0,0,0
                while(i3<=5):
                    if((tuplelist1[i3][0] == 'parent')&(tuplelist1[i3][2] == tuplelist1[n][2]]):
                        for j3 in range(6):
                            if( (tuplelist1[j3][0] == 'parent') & ((tuplelist1[i3][1] == tuplelist1[j3][1]) &
(tuplelist1[j3][2]!=tuplelist1[n][2]))):
                                for k3 in range(7):
                                    if((tuplelist2[k3][1] == tuplelist1[j3][2]) & (tuplelist2[k3][0] == 'male')):
                                        print(tuplelist1[j3][2]," ")

                                i3 = i3 + 1
                                m = m + 1

```

```

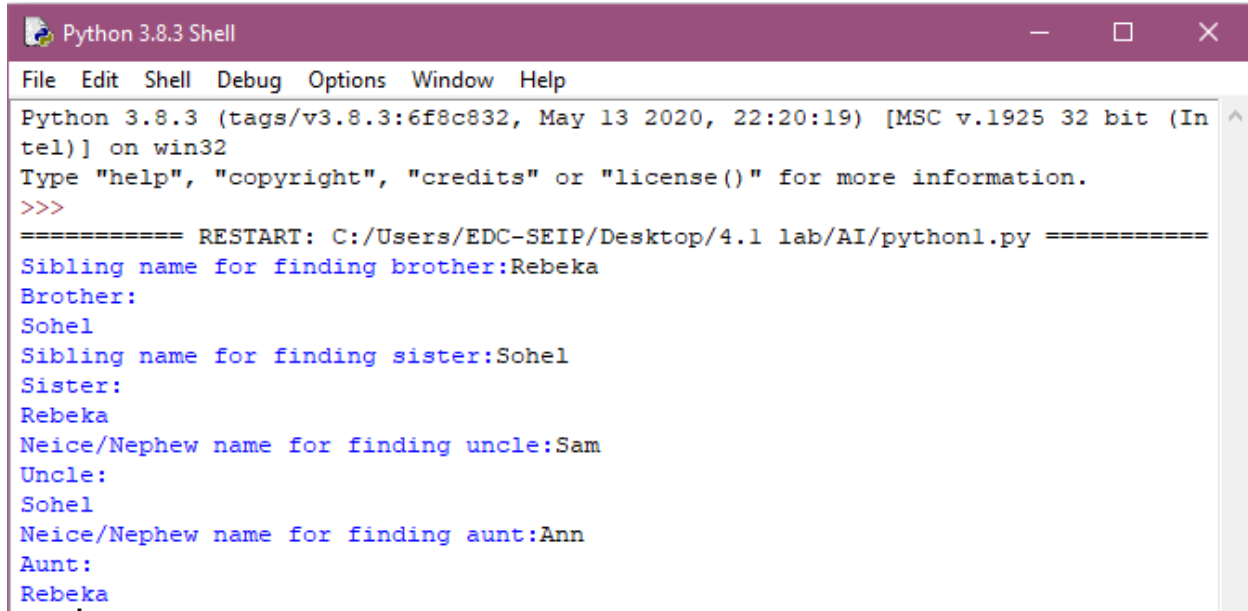
X4=str(input("Neice/Nephew name for finding aunt:"))
print('Aunt:')
q,r=0,0

while(q<=5):
    if((tuplelist1[q][0] == 'parent')&(tuplelist1[q][2] == X4)):
        for r in range(6):
            if( (tuplelist1[r][0] == 'parent') & (tuplelist1[q][1] == tuplelist1[r][2]) ):
                i4,j4,k4=0,0,0
                while(i4<=5):
                    if((tuplelist1[i4][0] == 'parent')&(tuplelist1[i4][2] == tuplelist1[r][2]]):
                        for j4 in range(6):
                            if( (tuplelist1[j4][0] == 'parent') & ((tuplelist1[i4][1] == tuplelist1[j4][1]) &
(tuplelist1[j4][2]!=tuplelist1[r][2]))):
                                for k4 in range(7):
                                    if((tuplelist2[k4][1] == tuplelist1[j4][2]) & (tuplelist2[k4][0] == 'female')):
                                        print(tuplelist1[j4][2]," ")

                                i4 = i4 + 1
                                q = q + 1

```

Python output:



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/EDC-SEIP/Desktop/4.1 lab/AI/python1.py =====
Sibling name for finding brother:Rebeka
Brother:
Soheli
Sibling name for finding sister:Soheli
Sister:
Rebeka
Neice/Nephew name for finding uncle:Sam
Uncle:
Soheli
Neice/Nephew name for finding aunt:Ann
Aunt:
Rebeka
```

Analyzing code and above input and output:

For finding brother/sister, we first determined the parent of the provided entry X1/X2 then we searched for other entries with similar parent. Later brother or sister was identified based on male and female properties. Similarly while finding uncle/aunt, firstly we determined the parent of X3/X4 and then searched for grandparents of X3/X4. Then the entries having parent similar to X3/X4's grandparents was chosen as the result.