Questions:

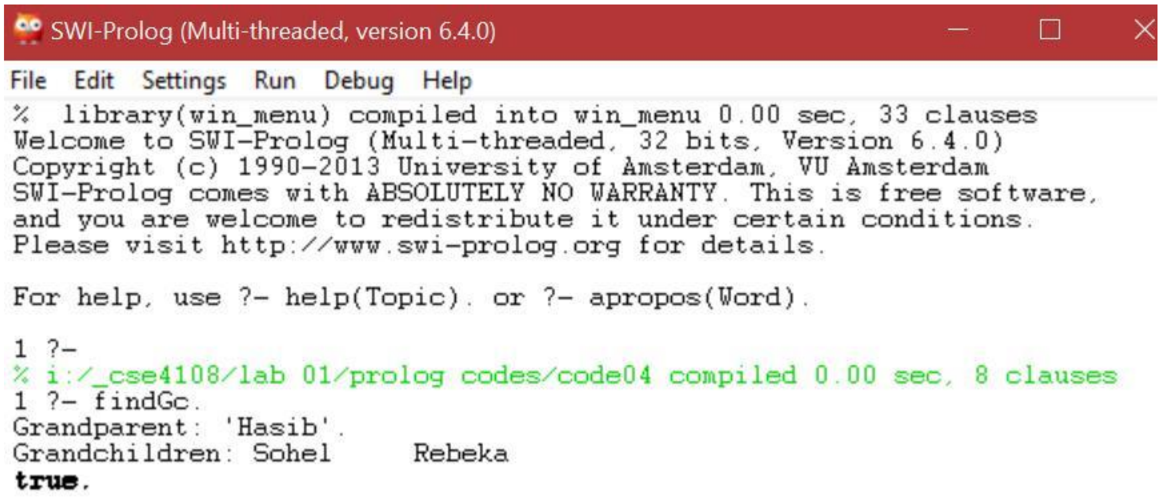
1. **Modify the demonstrated Python and Prolog codes to find the grandparents of somebody.**
2. **Enrich the demonstrated knowledge base with ‘brother’, ‘sister’, ‘uncle’ and ‘aunt’ rules in Python and Prolog.**

<u>Solution to the question no 1</u>

The demonstrated Prolog code to find grandchildren of someone is as below:

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


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


findGc:-
        write('Grandparent: '),read(X),write('Grandchildren:
        '), grandparent(X, Gc),write(Gc),tab(5),fail.
findGc.
```

A sample of input and output is as below:

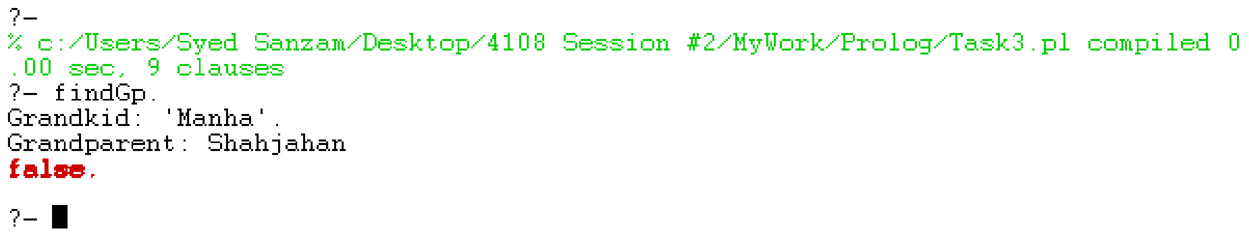

SWI-Prolog (Multi-threaded, version 6.4.0)
File Edit Settings Run Debug Help

```
% library(win_menu) compiled into win_menu 0.00 sec, 33 clauses
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 6.4.0)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?-
% i:/_cse4108/lab 01/prolog codes/code04 compiled 0.00 sec, 8 clauses
1 ?- findGc.
Grandparent: 'Hasib'.
Grandchildren: Sohel     Rebeka
true.
```

The aforementioned **Prolog** code has been modified as below so that for an input, grandparent(s) are displayed in response.

```prolog
parent('Shahjahan','Arnob').
parent('Shahjahan','Adit').
parent('Shahjahan','Anik').
parent('Shahjahan','Anna').
parent('Shahjahan','Amit').
parent('Shahjahan','Ashim').
parent('Ashim','Manha').



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


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

A sample of input and output is as below:

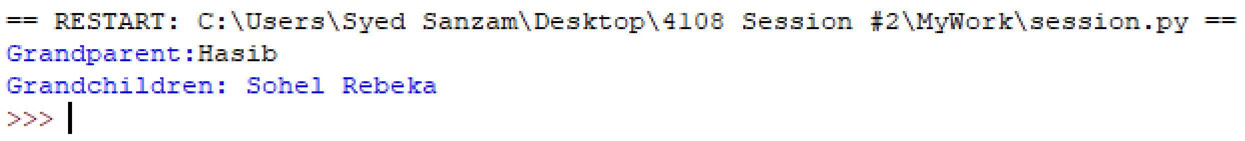


```
?-
% c:/Users/Syed Sanzam/Desktop/4108 Session #2/MyWork/Prolog/Task3.pl compiled 0
.00 sec, 9 clauses
?- findGp.
Grandkid: 'Manha'.
Grandparent: Shahjahan
false.

?- 
```

The demonstrated **Python** code to find grandchildren of someone is as below:

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

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

A sample of input and output is as below:

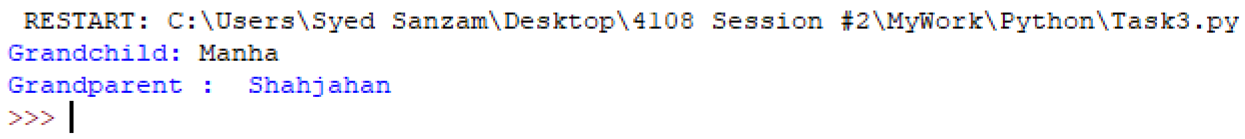


```
== RESTART: C:\Users\Syed Sanzam\Desktop\4108 Session #2\MyWork\session.py ==
Grandparent:Hasib
Grandchildren: Sohel Rebeka
>>> 
```

The aforementioned Python code has been modified as below so that for an input, grandparent(s) are displayed in response.

```python
tupleList1 = [('Parent','Shahjahan','Arnob'),
        ('Parent','Shahjahan','Anik'),
        ('Parent','Shahjahan','Arnob'),
        ('Parent','Shahjahan','Anna'),
        ('Parent','Shahjahan','Ashim'),
        ('Parent','Ashim','Manha')]

X = str(input("Grandchild: "))

print("Grandparent : ", end = ' ')
i = 0;
tupLen = len(tupleList1)
while(i < tupLen):
    if(tupleList1[i][0] == 'Parent')&(tupleList1[i][2] == X):
        par = tupleList1[i][1]
        for j in range(tupLen):
            if(tupleList1[j][0] == 'Parent') & (tupleList1[j][2] == par):
                grandpar = tupleList1[j][1]
                print(grandpar, end = ' ')
    i = i + 1;
```

A sample of input and output is as below:



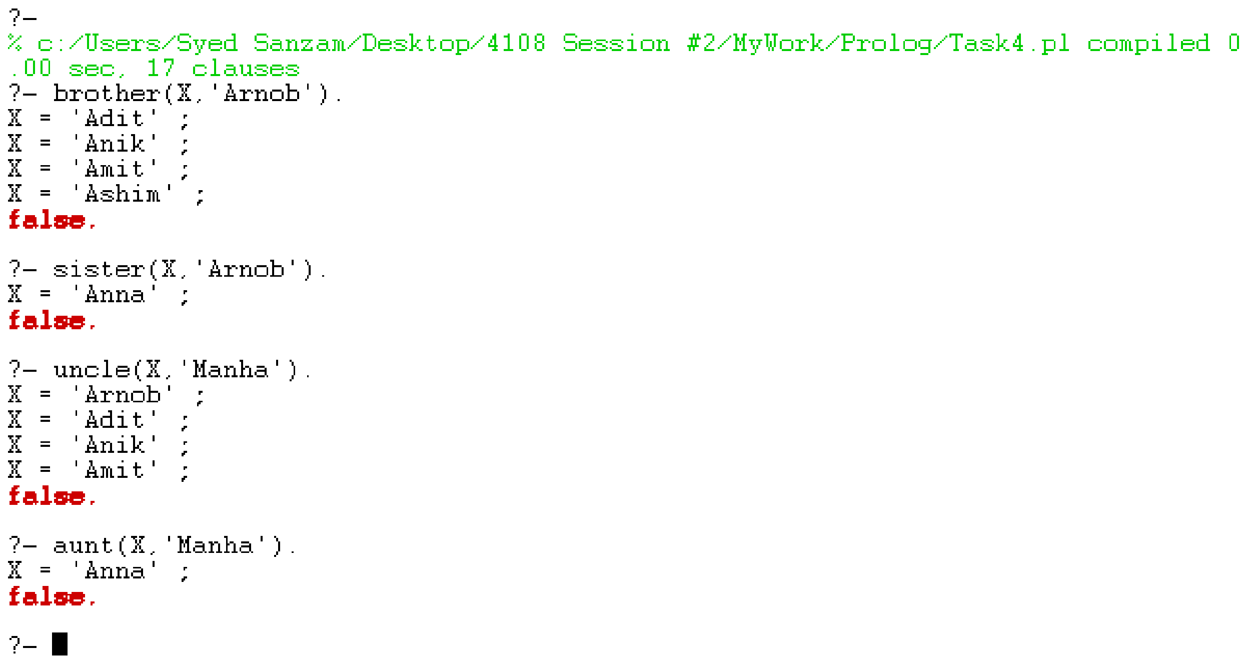
```
 RESTART: C:\Users\Syed Sanzam\Desktop\4108 Session #2\MyWork\Python\Task3.py
Grandchild: Manha
Grandparent :  Shahjahan
>>> 
```

<u>Solution to the question no 2</u>

The aforementioned **Prolog** code has been modified as below so that for an input, brother(s), sister(s), uncle(s) and aunt(s) are displayed in response.

```
parent('Shahjahan','Arnob').
parent('Shahjahan','Adit').
parent('Shahjahan','Anik').
parent('Shahjahan','Anna').
parent('Shahjahan','Amit').
parent('Shahjahan','Ashim').
parent('Ashim','Manha').


male('Arnob').
male('Anik').
male('Adit').
male('Ashim').
male('Amit').

female('Anna').

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


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

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


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

A sample of input and output is as below:

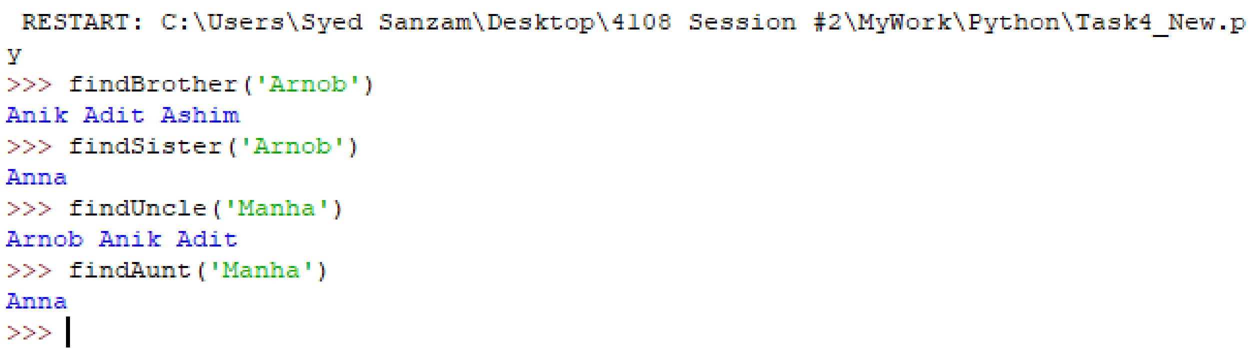


```
?-
% c:/Users/Syed Sanzam/Desktop/4108 Session #2/MyWork/Prolog/Task4.pl compiled 0
.00 sec, 17 clauses
?- brother(X,'Arnob').
X = 'Adit' ;
X = 'Anik' ;
X = 'Amit' ;
X = 'Ashim' ;
false.

?- sister(X,'Arnob').
X = 'Anna' ;
false.

?- uncle(X,'Manha').
X = 'Arnob' ;
X = 'Adit' ;
X = 'Anik' ;
X = 'Amit' ;
false.

?- aunt(X,'Manha').
X = 'Anna' ;
false.

?- 
```

The aforementioned **Python** code has been modified as below so that for an input, brother(s), sister(s), uncle(s) and aunt(s) are displayed in response.

```python
myTuple1 = [('Parent','Shahjahan','Arnob'),
        ('Parent','Shahjahan','Anik'),
        ('Parent','Shahjahan','Adit'),
        ('Parent','Shahjahan','Anna'),
        ('Parent','Shahjahan','Ashim'),
        ('Parent','Ashim','Manha')]


myTuple2 = [('Male','Shahjahan'),
        ('Male','Anik'),
        ('Male','Arnob'),
        ('Male','Adit'),
        ('Female','Anna'),
        ('Male','Ashim'),
        ('Female','Manha')]


totalTuple = len(myTuple1)

def findGender(X):
    tupleLen = len(myTuple2)
    for i in range(tupleLen):
        if(myTuple2[i][1] == X):
            return myTuple2[i][0]

def findBrother(X):
    i = 0
    found = 0

    while(i < totalTuple):
        if(myTuple1[i][0] == 'Parent') & (myTuple1[i][2] == X):
            par = myTuple1[i][1]
            for j in range(totalTuple):
                if(myTuple1[j][0] == 'Parent') & (myTuple1[j][1] == par):
                    if(myTuple1[j][2] != X) & (findGender(myTuple1[j][2]) == 'Male'):
                        bros = myTuple1[j][2]
                        print(bros, end = ' ')
        i = i + 1


def findSister(X):
    i = 0;
    while(i < totalTuple):
        if(myTuple1[i][0] == 'Parent') & (myTuple1[i][2] == X):
            par = myTuple1[i][1]
            for j in range(totalTuple):
                if(myTuple1[j][0] == 'Parent') & (myTuple1[j][1] == par):
                    if(myTuple1[j][2] != X) & (findGender(myTuple1[j][2]) == 'Female'):

                        sis = myTuple1[j][2]
                        print(sis, end = ' ')
        i = i + 1

def findUncle(X):
    i = 0
    while(i  < totalTuple):
        if(myTuple1[i][0] == 'Parent') & (myTuple1[i][2] == X):
            par = myTuple1[i][1]
            findBrother(par)
        i = i + 1

def findAunt(X):
    i = 0
    while(i  < totalTuple):
        if(myTuple1[i][0] == 'Parent') & (myTuple1[i][2] == X):
            par = myTuple1[i][1]
            findSister(par)
        i = i + 1
```

A sample of input and output is as below:



```
 RESTART: C:\Users\Syed Sanzam\Desktop\4108 Session #2\MyWork\Python\Task4_New.p
y
>>> findBrother('Arnob')
Anik Adit Ashim
>>> findSister('Arnob')
Anna
>>> findUncle('Manha')
Arnob Anik Adit
>>> findAunt('Manha')
Anna
>>> 
```