

Homework 3

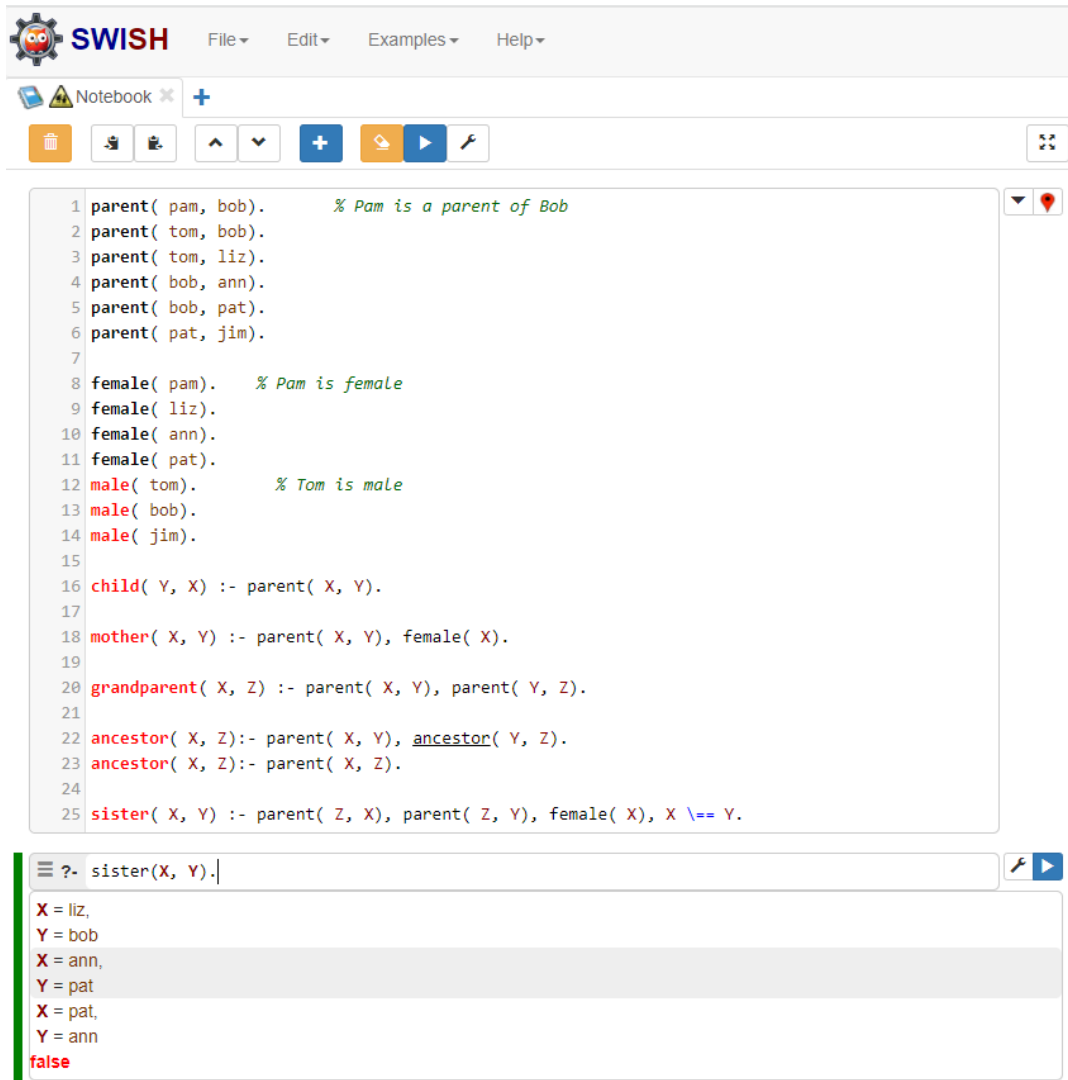
FULYA KOCAMAN

CWID: 803023878

Due: Check date on Canvas. Prepare your answers as a **single PDF file**.

1. The attached `family.pl` Prolog program describes a set of facts and rules to identify some family relationships. `parent(pam, bob)` means that pam is the parent of bob.
 - a. Add one new rule to `family.clp` to print a list of all (sister, person) pairs. X is a sister of Y if they have the same parent and X is female. [Show rule and output of the program]

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



```
1 parent( pam, bob).      % Pam is a parent of Bob
2 parent( tom, bob).
3 parent( tom, liz).
4 parent( bob, ann).
5 parent( bob, pat).
6 parent( pat, jim).
7
8 female( pam).          % Pam is female
9 female( liz).
10 female( ann).
11 female( pat).
12 male( tom).           % Tom is male
13 male( bob).
14 male( jim).
15
16 child( Y, X) :- parent( X, Y).
17
18 mother( X, Y) :- parent( X, Y), female( X).
19
20 grandparent( X, Z) :- parent( X, Y), parent( Y, Z).
21
22 ancestor( X, Z):- parent( X, Y), ancestor( Y, Z).
23 ancestor( X, Z):- parent( X, Z).
24
25 sister( X, Y) :- parent( Z, X), parent( Z, Y), female( X), X \== Y.
```

?- sister(X, Y).|

```
X = liz,
Y = bob
X = ann,
Y = pat
X = pat,
Y = ann
false
```

- b. Add one new rule to family.clp to print a list of all (aunt, nephew/niece) pairs. [Show rule and output of the program]

aunt(X, Y) :- sister(X, Z), parent(Z, Y). %X is an aunt to Y

The image shows the SWISH Prolog IDE interface. The top menu bar includes 'File', 'Edit', 'Examples', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main editor area contains a Prolog program with the following rules and facts:

```
1 parent( pam, bob).      % Pam is a parent of Bob
2 parent( tom, bob).
3 parent( tom, liz).
4 parent( bob, ann).
5 parent( bob, pat).
6 parent( pat, jim).
7
8 female( pam).          % Pam is female
9 female( liz).
10 female( ann).
11 female( pat).
12 male( tom).           % Tom is male
13 male( bob).
14 male( jim).
15
16 child( Y, X) :- parent( X, Y).
17
18 mother( X, Y) :- parent( X, Y), female( X).
19
20 grandparent( X, Z) :- parent( X, Y), parent( Y, Z).
21
22 ancestor( X, Z):- parent( X, Y), ancestor( Y, Z).
23 ancestor( X, Z):- parent( X, Z).
24
25 sister( X, Y) :- parent( Z, X), parent( Z, Y), female( X), X \== Y. %X is a si
26
27 aunt( X, Y) :- sister( X, Z), parent( Z, Y). %X is an aunt to Y
28
```

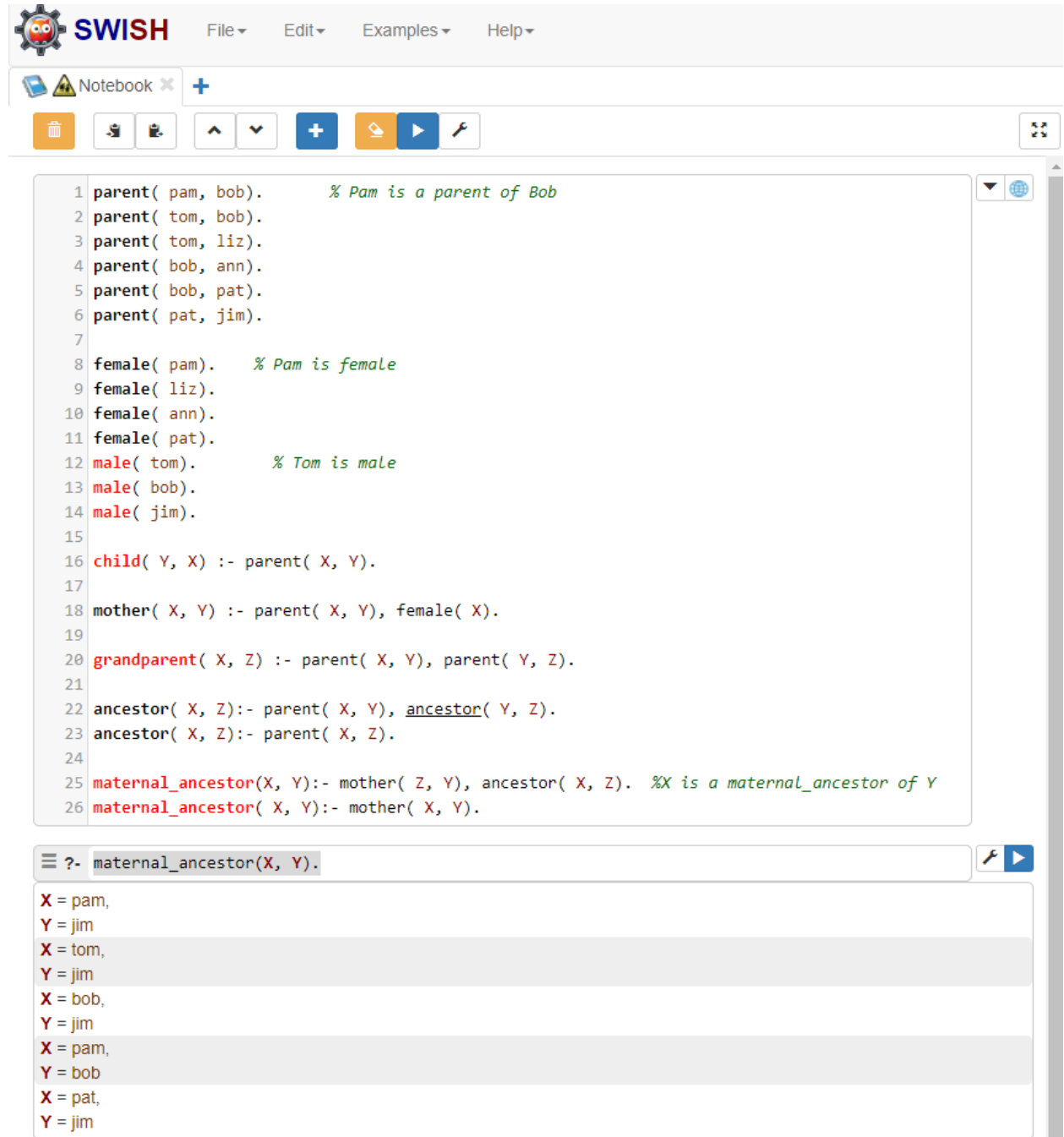
Below the editor, the query 'aunt(X, Y).' is entered in the console. The output shows the following results:

```
X = liz,
Y = ann
X = liz,
Y = pat
X = ann,
Y = jim
false
```

- c. Add one new rule to family.clp to print a list of all (maternal_ancestor, person) pairs. A maternal ancestor is an ancestor via the person's mother's side, e.g., mother, mother's father, mother's grandfathers, mother's grandmothers, ... [Show rules and output of the program]

maternal_ancestor(X, Y):- mother(Z, Y), ancestor(X, Z). %X is a maternal_ancestor of Y

maternal_ancestor(X, Y):- mother(X, Y).



The screenshot shows the SWISH Prolog IDE interface. The top menu bar includes 'File', 'Edit', 'Examples', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main editor area contains a Prolog program with the following rules and facts:

```
1 parent( pam, bob).           % Pam is a parent of Bob
2 parent( tom, bob).
3 parent( tom, liz).
4 parent( bob, ann).
5 parent( bob, pat).
6 parent( pat, jim).
7
8 female( pam).               % Pam is female
9 female( liz).
10 female( ann).
11 female( pat).
12 male( tom).                % Tom is male
13 male( bob).
14 male( jim).
15
16 child( Y, X) :- parent( X, Y).
17
18 mother( X, Y) :- parent( X, Y), female( X).
19
20 grandparent( X, Z) :- parent( X, Y), parent( Y, Z).
21
22 ancestor( X, Z):- parent( X, Y), ancestor( Y, Z).
23 ancestor( X, Z):- parent( X, Z).
24
25 maternal_ancestor(X, Y):- mother( Z, Y), ancestor( X, Z). %X is a maternal_ancestor of Y
26 maternal_ancestor( X, Y):- mother( X, Y).
```

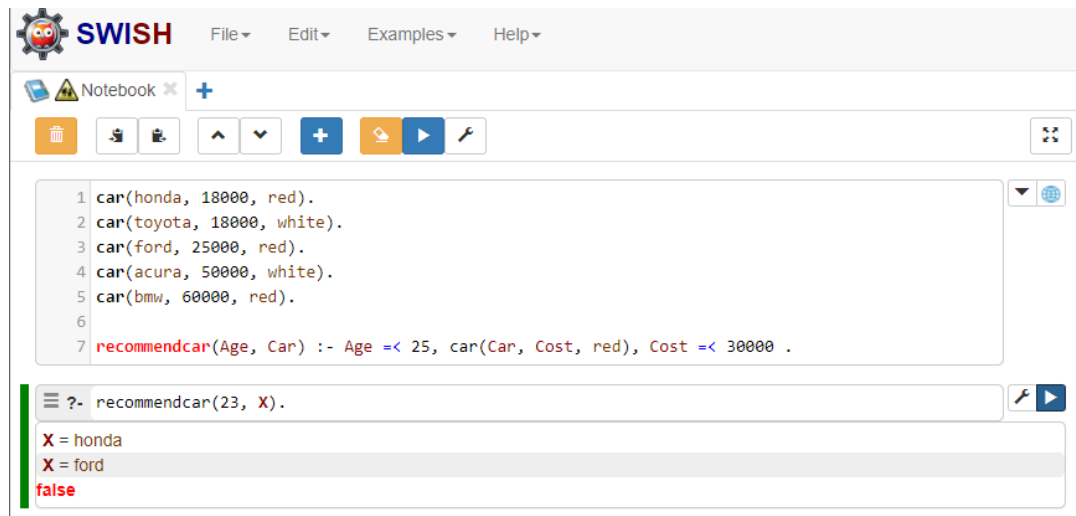
Below the editor, a query is entered in the 'Query' field: `?- maternal_ancestor(X, Y).` The results are displayed in a list:

```
X = pam,
Y = jim
X = tom,
Y = jim
X = bob,
Y = jim
X = pam,
Y = bob
X = pat,
Y = jim
```

2. The attached `cars.pl` Prolog program describes a list of cars (brand, price, color) and a rule that recommends a car that costs less than \$30,000 for a person younger than 25 years of age [same problem as in the CLIPS question from the last homework]. Note that you “run” this program by posing a query: `recommendcar(23, X)`.

- a. Modify the rule such that the recommendation for a person younger than 25 years is a car that costs less than \$30,000 and red in color. [Show modified rule and output of the program]

`recommendcar(Age, Car) :- Age <= 25, car(Car, Cost, red), Cost <= 30000 .`



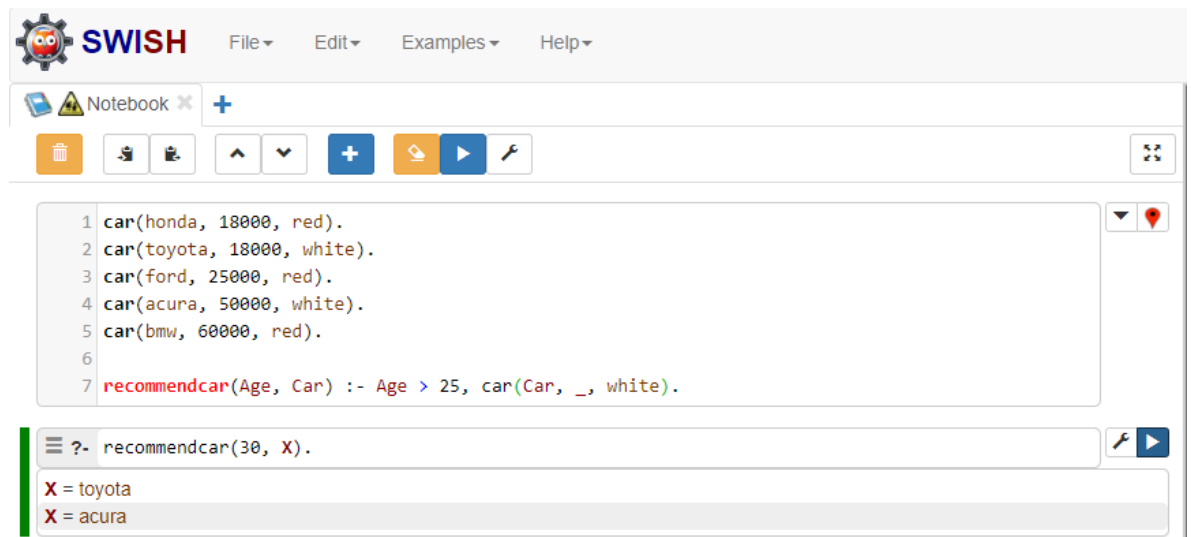
The screenshot shows the SWISH Prolog IDE. The code editor contains the following Prolog code:

```
1 car(honda, 18000, red).
2 car(toyota, 18000, white).
3 car(ford, 25000, red).
4 car(acura, 50000, white).
5 car(bmw, 60000, red).
6
7 recommendcar(Age, Car) :- Age <= 25, car(Car, Cost, red), Cost <= 30000 .
```

The query `?- recommendcar(23, X).` is entered in the query box. The output shows two solutions: `X = honda` and `X = ford`, followed by `false`.

- b. Change the program such that the recommendation for a person older than 25 years is a white car. [Show rules and output of the program]

`recommendcar(Age, Car) :- Age > 25, car(Car, _, white).`



The screenshot shows the SWISH Prolog IDE with the modified Prolog code:

```
1 car(honda, 18000, red).
2 car(toyota, 18000, white).
3 car(ford, 25000, red).
4 car(acura, 50000, white).
5 car(bmw, 60000, red).
6
7 recommendcar(Age, Car) :- Age > 25, car(Car, _, white).
```

The query `?- recommendcar(30, X).` is entered in the query box. The output shows two solutions: `X = toyota` and `X = acura`.