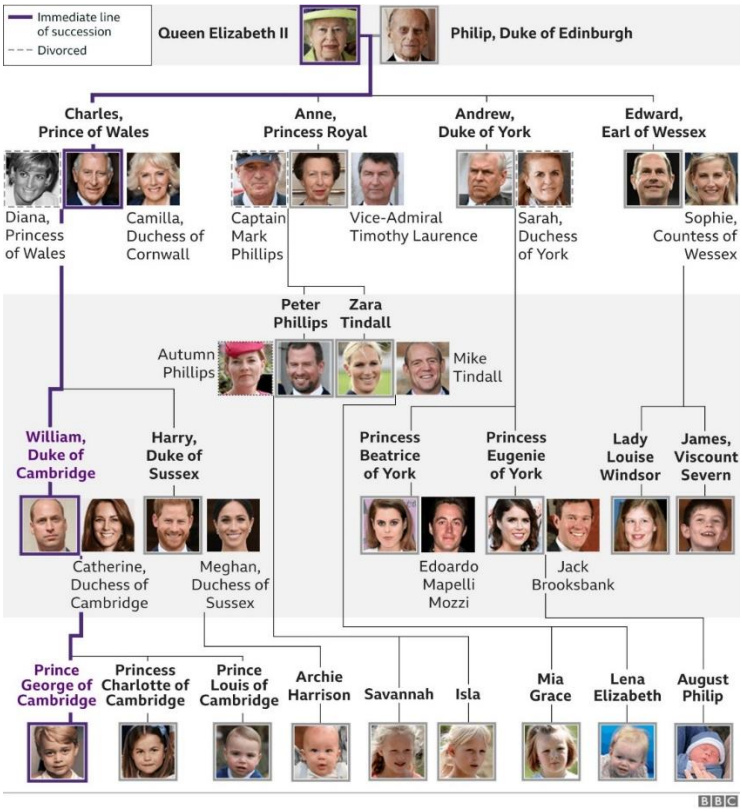
Artificial Intelligence

ASSIGNMENT – 2

Warning: We will use random generator to pick and call 10 people. We will give them a simple prolog problem to solve, right in front of us (i.e. share your screen). If you do the below assignments by yourself, it should take you 3 minutes or less to solve it. If you fail you will receive 0 for this assignment. Of course, in addition, you will receive an additional penalty of 10 towards your overall grade. The chances of you getting caught for plagiarism is 1/20. But I strongly recommend against rolling the dice.

Instructions: (Read very carefully)

- For this assignment [20 Marks], you will create facts and define rules in a single prolog file. **Submit the prolog file (.pl).**
  - Name the file as follows: S2018xxxxxxxx_assign2.pl
- From the swi-prolog interpreter, run queries for every single rule (in total, there are 10 rules). Take screenshot(s) of the output as .PNG or .JPG file(s).
  - Name the file as follows: S2018xxxxxxx_ouput1.png
- The output of the queries may be too big for a single-screenshot-image. In that case, its ok take two screenshots, one for the first 5 rules and the second for the remaining 5.
- Any deviation in submission format or naming conventions i.e., submitting .txt, .docx or .pdf files will result in zero marks.



Representing the family tree using facts [2 Marks]

For each node in the family tree, you have four pieces of information

1. The name of the person
    a. Represent this information using the predicate:
    name(<short-name>, <full-name>)
    example: name(edoardo, "Edoardo Mapelli Mozzi")

    Notes:
    1. The string for the full-name can only appear in facts of 'name' predicate. From this point on you should only use the short-form for the remaining facts.
    2. Sometimes, the name itself is short, example "Isla". But still, you must use the above predicate to represent the names as name(isla, "Isla").
    3. Ensure that all short-names are unique. You can shorten as you like. For example, you can shorten "Timothy Laurence" as tim, timothy, or laurence.
2. The gender of the person
    a. There are two options to represent this. You may choose the one that suits your taste
    b. Option 1: You can use two predicates male(<short-name>) and female<short-name>)
    example: male(william)

    c. Option 2: You can use a single predicate: gender(<short-name>,<gender>) example:
    example: gender(isla, female)

3. The parent(s) of the person
    a. Use the predicate parent(<parent-short-name>, <child-short-name>) to represent parent-child relationships.
    example: parent(diana, william)

4. The noble-title/rank of the person (only if it exists)
    a. Use the predicate position(<short-name>, <rank>, <place>) to represent this information.
    example: position(meghan, duchess, sussex).

    Notes:
    1. If the place name is not specifically mentioned, use 'england' as the place name. For example 'Captain Mark Phillips' should be represented as position(mark, captain, england)
    2. Sometime the rank appears first. For instance 'Lady Louis Windsor" should be represented as position(louis, lady, windsor).

Rules to define relations and connections

Define the below prolog rules to answer queries about connections

1. father(X, Y) is true if X is the father of Y. [1 Mark]
    example: father(peter,isla) should return true

2. mother(X,Y) is true if X is the mother of Y [1 Mark]
    example: mother(autumn, isla) should return true

3. siblings(X,Y) is true is X is a sibling of Y. [1 Mark]
    example: sibling(isla, savannah) should return true.

4. Descendant(X,Y) is true if X is a descendant of Y. [2 Marks]

example: descendant(isla, elizabeth) should return true

5. firstcousin(X, Y) is true if X is a first-cousin of Y. [2 Marks]
   example: firstcousin(isla,mia) should return true

6. cousin(X,Y) is true if X and Y are in the same level/depth of the tree. [2 Marks]
   example: cousin(august, george) should return true

7. connection(X,Y,L) where L is a list of people in between X and Y. [2 Marks]
   example: connection(isla, mia, L) should return

   L = [isla, peter, zara, mia]

   Because Isla's father is Peter and Peter's sister is Zara and Zara's daughter is mia
   Note: Some of the predicates above (example: firstcousin) are symmetric in nature

---

Now Let us learn a little bit about the rank and hierarchy of the nobles.

Emperor	Empress
King	Queen
Grand Duke	Grand Duchess
Grand Prince	Grand Princess
Archduke	Archduchess
Duke	Duchess
Prince	Princess
Marquess	Marchioness
Count/Earl	Countess
Viscount	Viscountess
Baron	Baroness
Knight	Dame
Lord	Lady

Titles in the same row have the same rank. For instance, Baron and Baroness are of equal rank except for their gender. Titles in the rows below have lower rank. For example, prince/princess is lower rank to Duke/Duchess.

Facts for ranks [1 Mark]

1. Represent equal ranks using the predicate 'equalrank(<rank>)'
   example: equalrank(emperor, empress)

2. Represent lower ranks using the predicate 'lowerrank(<lowerrank>,<higherrank>).
   example: lowerrank(king, emperor)

Rules to use the rank to answer the following

1. lowerthan(X, L) is true if L is a list of people who are of lower rank than X. [2 Marks]
2. higherthan(X, L) is true if L is a list of people who are of higher rank than X. [2 Marks]
3. sameas(X,L) is true if L is a list of people who are the same rank as X. [2 Marks]

Hints and directions

1. For the rules to work, you might want to add facts for civilians (those who doesn't have any rank). For example, 'Isla' does not have any rank. So, you can add this as a fact: position(isla, civilian, england).
2. Likewise, even though not explicitly mentioned in the hierarchy. You can add a fact to establish 'civilian' as the lowest of all ranks.
3. All predicates should function as follows.
   a. If queried for father(peter, isla) it should return true.
   b. If instead queried for father(X, isla), then it should return X=peter.
   c. If queried for father(peter,X), then it should return X= isla; X= savannah.
   d. If queried for father(X,Y) it should return all father-child pairs one after the other
   e. The same goes for predicates which has 'list' as argument. If an empty (unbound) variable is passed for a list, a populated list should be returned.

Marks breakdown

1. Representing facts about the family tree gets 2 Marks
2. Rules for connections 11 Marks
   a. Simple rules (1,2,3) get 1 Mark each
   b. Slightly complex ones (4,5,6,7) get 2 Marks each
3. Representing facts about the ranks gets 1 Mark
4. Rules for ranks gets 2 Marks each for the 3 rules, so 6 marks in total.

There might be a few of you who may not understand 'cousin' vs 'firstcousin'. Note: 'cousin' is everyone in orange except for the sibling (all of them in the same level of the tree).