TASK 4 Mistakes:

There were no major problems with first three sub tasks of the Task 4-2. But some of you have faced the problem with the fourth and fifth. Here you may find some of the right answers to the tasks:

Task 4-2.4:

The rule for CWM is:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.smith-family.com/familyOntology.owl#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .

{?a ex:hasChild ?b, ?c .
 ?b ex:hasChild ?x .
 ?c ex:hasChild ?y .
 ?b log:notEqualTo ?c .
} => {?x ex:hasCousin ?y} .
```

the result is:

```
@prefix : <#> .
@prefix ex: <http://www.smith-family.com/familyOntology.owl#> .

    :charles     ex:hasCousin :john .
    :john     ex:hasCousin :charles, :robert .
    :robert     ex:hasCousin :john .
```

If it is more convenient for someone to operate with *hasParent* property, you may extend the solution with extra rule to define this relation (see below).

The SWRL rule is:

```
hasChild(?a, ?b) -> hasParent(?b, ?a) .
hasParent(?a, ?b), hasParent(?b, ?e), hasParent(?c, ?d),
hasParent(?d, ?e),  DifferentFrom (?b, ?d) -> hasCousin(?a, ?c), hasCousin(?c, ?a) .
```

or

```
hasChild(?ch1, ?cousin1), hasChild(?ch2, ?cousin2),
hasChild(?par1, ?ch1), hasChild(?par1, ?ch2),
DifferentFrom (?ch1, ?ch2) -> hasCousin(?cousin1, ?cousin2), hasCousin(?cousin2,
?cousin1).
```

Tip: Be sure that your instances are different from each other - marked as "Different instances" in the ontology.

To compare URIs you may use *DifferentFrom()* atom. Taking into account "non-unique name" assumption of OWL, several different names may represent the same entity. Therefore, we have to explicitly specify if the names represent different entities.

Examples above present cousins "in blood". If we would like to take into account "Step-Cousin", we have to modify the rule. For example, via such extension:

```
hasWife(?h,?w) -> hasSpouse(?h,?w), hasSpouse(?w,?h).
hasHusband(?w,?h) -> hasSpouse(?h,?w), hasSpouse(?w,?h).
hasChild(?pch3, ?pcousin1), hasChild(?pch2, ?pcousin2),
hasChild(?par1, ?pch1), hasSpouse(?pch1, ?pch3), hasChild(?par1, ?pch2),
DifferentFrom (?pch1, ?pch2) -> hasCousin(?pcousin1, ?pcousin2), hasCousin(?pcousin2,
?pcousin1).
```

the result will include:

```
:mary     :hasCousin :charles, :robert .
:charles  :hasCousin :mary .
:robert   :hasCousin :mary .
```

Task 4-2.5: **Is even more tricky task…** According to the Open World Assumption, if we do not have certain fact in our dataset, it does not mean that it is false. It means that we just do not have any knowledge about it. It might be true or false. Therefore, there is no way to reason about a fact which does not exist. The only way to solve the problem is to imitate a Close World. With *log:semantics* built-in we may associate out dataset with some variable and check whether this graph (set of triples) includes some triple or not (*log:notIncludes* built-in).

Note: I am not going to reduce the grade for those who tried, but did not find right solution for this task.

Built-ins of CWM allow imitating Close World Assumption. The rule for CWM is:

```
@prefix ex: <http://www.smith-family.com/familyOntology.owl#>.
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.

{?a ex:hasWife ?b . ?b ex:hasHusband ?a} => {?a ex:hasSpouse ?b . ?b ex:hasSpouse ?a}.
{?y ex:hasChild ?x . ?y ex:hasSpouse ?b . <dataNew.n3> log:semantics ?t . ?t
log:notIncludes {?b ex:hasChild ?x}} => {?b ex:hasStepChild ?x} .
```

the result is:

```
@prefix : <http://www.smith-family.com/familyOntology.owl#> .
@prefix dat: <#> .
       …
dat:bill     a :LivingPerson,
               :Person;
               :hasChild dat:john;
               :hasSpouse dat:kate;
               :hasStepChild dat:mary;
               :hasWife dat:kate .
       …
```

The SWRL rule is:

Due to the open world assumption, it is impossible to achieve the result with the current data. There is no possibility to imitate Close World in Protégé.

Please be aware, that if you use blank node definition in the rule, reasoner searches for a corresponding pattern with blank node. So, if the dataset does not contain blank node, the pattern from the rule will not be matched.