Task 1 Mistakes:

Task 1-1

a) ***Do not mix-up*** between capital and small letters in the names.

b) Some of you made a mistake while define the prefixes… If the whole mane is `<http://www.smith-family.com/family#Peter>`, and you would like to use qualified name as `f:Peter`, then the prefix should be defined as:
@prefix f: <http://www.smith-family.com/family#> .
not the @prefix f: <http://www.smith-family.com/family/> .
not the @prefix f: <http://www.smith-family.com/family> .
Do not forget to wrap value of the prefix to (< >).
Do not forget to leave a "white space" between prefix name and its value.

c) Some of you tried to use ***@base*** abbreviation and totally mixed up relative URI with *qnames* (that use prefix-based definition). Relative URI should be present inside < > assuming that first part comes from @base definition. It is also possible to use empty prefix (:).

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

<mary> rdf:type <Person> .
<john> rdf:type foaf:Person .
:jj rdf:type :Dog .

:Person rdf:type rdfs:Class.
:Dog rdf:type rdfs:Class.
```

d) Do not use any prefixes before ***Literals***:
`x:Peter x:hasAge x:"67".`
`the correct way is: x:Peter x:hasAge "67".`

e) Some of you did not separate ***predicate*** and ***object*** with a "white space" in case when ***object is literal***:
`x:Peter x:hasAge"67".`
`the correct way is: x:Peter x:hasAge "67".`

f) ***Do not forget*** to wrap ***Literals*** into double quotes (" ").
g) ***Do not forget*** to include ***prefix*** definition to the rdf, in case you use qualified names.
h) ***Do not forget*** to include ***prefix*** before the rest (last part) of qualified name.
i) ***Do not make*** any spaces between the ***prefix*** and the rest of qualified name.
j) ***Do not forget*** the (.) after each statement (triple) and after prefix definition as well.

Task 1-2

a) Many mistakes from the Task-1 are also made in Task-2.

b) Some of you forgot to include "/" in the end of prefix definition:
If the whole mane is `<http://pizza-larry.fi/menu/pizzaCardinal>`, and you would like to use qualified name as `m:pizzaCardinal`, then the prefix should be defined as:
@prefix m: <http://pizza-larry.fi/menu/> .
not the @prefix m: <http://pizza-larry.fi/menu> .

c) In case you would like to use type definition for ***Literals***, you must use full name of the type or its qualified name with correspondent prefix. See example:
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
…
menu:PizzaLake buz:hasPrice "6.50"^^xsd:float .
not the menu:PizzaLake buz:hasPrice "6.50"^^float .

Do not make any spaces between value of the ***Literal*** and its type:
not the menu:PizzaLake buz:hasPrice "6.50" ^^xsd:float .

d) The purpose of semantic sugar comma "`,`" is not the same as abbreviation "`()`" (used for RDF Collection). RDF Collection is a possibility to define a list in RDF that has a structure [first, rest], where *rest* has the same structure [first, rest]. So, Collection looks like [first,[first,[first, …]]] - is a ordered list. In case you use Collections, remember that members of Collection are separated by a "white space" (not a comma). More about "`()`" abbreviation and RDF Collection, you can find in Turtle documentation.

e) Some of you have used whole name of the ***property*** (predicate of the statement/triple) to define certain prefix, and then have used this prefix as a prefix for the ***object*** of the statement/triple. See example:
@prefix t: <http://www.food.com/ontology/hasTopping> .
m:pizzaCardinal t:shrimp, t:ham .
As you can see, the predicate (property) of the triple is missing. ***t:*** here is a prefix of ***shrimp*** and ***ham*** objects. The right way to write the triple in this case is:
@prefix t: <http://www.food.com/ontology/hasTopping> .
@prefix s: <http://www.food.com/ontology/> .
m:pizzaCardinal t: s:shrimp, s:ham . Where ***s:*** is correspondent prefix for ***shrimp*** and ***ham*** objects, and ***t:*** is a predicate (property) of the statement/triple without the rest part of qualified name (because prefix contains whole URI/name of the property).

f) Some of you have tried to write statements about resource by going "in depth" by graph structure:
`x:about p:sells y:pizzaCardinal z:hasTopping z:shrimp, z:ham; p:hasPrice"5.50".`
If you would like to write several statements about the same ***subject***, then you may combine several triples using ";" and "," abbreviations. But, you should remember that you describe the same ***subject*** and should describe only the properties of this particular subject (describe only those edges of the graph that originate only from this ***subject***). In the presented example, part `z:hasTopping z:shrimp, z:ham; p:hasPrice"5.50".` belongs to another ***subject*** (belongs to `y:pizzaCardinal`, not to `x:about`).

g) Some of you asked whether it is possible to use prefixes to define another prefixes. In case we use already defined prefix (not the prefix that will be defined later) to define another one, parser may accept it.

```
@prefix m: <http://pizza-larry.fi/menu/> .
@prefix p: m:pizza .
@prefix s: <http://www.food.com/ontology/> .
m:Cardinal s:hasTopping s:shrimp, s:ham .
```

But the problem is - parser does not accept some of the characters (e.g. "/", "#") used in qualified names. So, it will not work for:

```
@prefix l: <http://pizza-larry.fi/> .
@prefix m: l:menu/ .
@prefix s: <http://www.food.com/ontology/> .
m:pizzaCardinal s:hasTopping s:shrimp, s:ham .
```

or

```
@prefix l: <http://pizza-larry.fi/> .
@prefix m: l:menu .
@prefix s: <http://www.food.com/ontology/> .
m:/pizzaCardinal s:hasTopping s:shrimp, s:ham .
```

h) The main idea about graphical representation of RDF is that each arrow (together with nodes it comes from and refers to) represents a triple (subject-predicate-object). Therefore, amount of triples should be at least equal to the numbers of arrows. In case of reification, the object of the triple becomes a Statement (which is a triple itself) - resource with is an instance of rdf:Statement class. So, definition of the statement itself consists of at least 4 triples.

i) Correct solutions for reification are following:

```
@prefix st: <http://pizza-larry.fi/staff/> .
@prefix b: <http://business.com/ontology/> .
@prefix m: <http://pizza-larry.fi/menu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
st:manager b:defines _:p1.
_:p1 rdf:type rdf:Statement.
_:p1 rdf:subject  m:pizzaCardinal.
_:p1 rdf:predicate b:hasPrice.
_:p1 rdf:object "5.50".
…
```

or

```
…
st:manager b:defines [ rdf:type rdf:Statement; rdf:subject  m:pizzaCardinal; rdf:predicate b:hasPrice;
rdf:object "5.50"]
…
```

It is also possible to use some other concrete names instead of blank nodes.

```
…
@prefix p: <http://pizza-larry.fi/price/> .
st:manager b:defines p:p1.
p:p1 rdf:type rdf:Statement; rdf:subject  m:pizzaCardinal; rdf:predicate b:hasPrice; rdf:object "5.50".
…
```

Some of you did not directly included triples that state prices for pizzas, assuming that they are included into Statement definitions. Therefore, this information about prices is not explicitly present in description. It could be inferred from the Statement definition in case if storage supports such inference. It is better to explicitly include those triples to be absolutely sure.

Be aware that all the names are *case sensitive*!

Task 1-3

a) If we have a blank node *[]* with some descriptions of its properties, than on the graph, we draw it as an empty box (without a name) with the arrows for the correspondent properties of this blank node. Example:
b:hp1 b:isAbout [a a:Sorcerer] .

b) ***Do not forget*** to include ***prefix*** definition to the graphical representation of the rdf, in case you use qualified names. Even if some prefixes were defined in initial RDF representation format, graphical representation should also have corresponding prefix definitions.

c) ***Do not forget*** to wrap Literal values in (“”) or use different graphical box (or other element) that differs from URI representation, otherwise these values are considered as URIs.

d) ***Do not use*** **<>** in resource names in graphical representation. The URI or full name of a resource is wrapped with **<>** in *n-triples* notation (and some other notations). **<>** are not a part of the resource names.

e) ***a*** is an abbreviation for ***<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>*** or ***rdf:type*** if we use qualified name with correspondent definition for ***rdf:*** prefix.
not for rdf or ***<http://www.w3.org/1999/02/22-rdf-syntax-ns#>***
not for rdf:a or ***<http://www.w3.org/1999/02/22-rdf-syntax-ns#a>***