Sample Solutions for Coursework C1: XPath

These are sample solutions for the queries in coursework C1. There are usually many different ways of expressing a query and this list shows one possible formulation in XPath for each query on the task list.

1. How many calories are in Linguine alla Pescatora?

```
//recipe[title="Linguine alla Pescatora"]/nutrition/@calories
```

2. Return the titles of the recipes that have more than 500 calories.

```
//recipe[nutrition/@calories > 500]/title
```

3. Return the recipes for which at least 4 eggs are needed.

```
//recipe[.//ingredient[@name="eggs" and @amount >= 4]]
```

4. Which recipe has the highest number of calories? (Do not use the XQuery function `max`!)

```
//recipe[not(nutrition/@calories < //nutrition/@calories)]
```

5. How many ingredients are there in Ricotta Pie?

```
count(//recipe[title="Ricotta Pie"]//ingredient)
```

6. How many compound ingredients (i.e., ingredients with ingredients) are there in Ricotta Pie?

```
count(//recipe[title="Ricotta Pie"]//ingredient[ingredient])
```

7. How many elementary (= non-compound) ingredients are there in Ricotta Pie? (An ingredient is elementary if it does not have ingredients itself.)

```
count(//recipe[title="Ricotta Pie"]
                    //ingredient[not(ingredient)])
```

8. Which recipes have an ingredient whose preparation needs more steps than are needed for the recipe itself (i.e., top level steps)?

```
//recipe[ingredient[count(preparation/step)
              > count(ancestor::recipe/preparation/step)]]
```

9. What is the average number of calories per recipe? (Do not use the XQuery function `avg`!) Note: Since the "/" operator has its own meaning in XPath, the division operator is infix `div`.

```
sum(//recipe/nutrition/@calories) div count(//recipe)
```

10. Return the names of the ingredients of Zuppa Inglese.

```
//recipe[title="Zuppa Inglese"]//ingredient/@name
```

11. Return the names of those ingredients of Zuppa Inglese that occur also in other recipes.

```
//recipe[title="Zuppa Inglese"]
   //ingredient[@name =
                //recipe[title!="Zuppa Inglese"]
                   //ingredient/@name]
      /@name
```

12. Which recipes have an ingredient in common with Zuppa Inglese?

```
//recipe[.//ingredient/@name =
         //recipe[title="Zuppa Inglese"]//ingredient/@name]
```

13. Return the ingredients of recipes other than Zuppa Inglese that these recipes have in common with Zuppa Inglese.

```
//recipe[title!="Zuppa Inglese"]
   //ingredient[@name =
                //recipe[title="Zuppa Inglese"]
                   //ingredient/@name]
```

14. Return the names of all elementary ingredients that occur in at least two recipes.

```
//ingredient[not(ingredient)]
            [@name =
             ancestor::recipe/following::ingredient/@name]
   /@name
```

15. Return the titles of all recipes for which some form of egg is needed (like "egg whites" or "egg yolk").

```
//recipe[.//ingredient[contains(@name,"egg")]]/title
```

16. Return the titles of the recipes that have only elementary ingredients.

```
//recipe[not(ingredient/ingredient)]/title
```

17. Return the names of those ingredients that are mentioned in a preparation step of their recipe.

    I did not find a fully correct solution and it may even be that it cannot be expressed in XPath (1.0). I came up first with the following one:

```
//ingredient[contains(ancestor::recipe//step,@name)]/@name
```

    Explain why this misses answers! Then I improved as follows:

```
//ingredient[contains(ancestor::recipe//preparation,
                    @name)]/@name
```

    Explain why this is still incorrect! Last year, a student came up with this one:

```
//ingredient[contains(following-sibling::preparation,
                    @name)]/@name
```

    Do you see that this is better, but still misses some answers?

18. Return the names of ingredients that are not mentioned in a preparation step of their recipe.

    I give a sample solution based on the last "solution" to the preceding question. As that one is not fully correct, the one below is not either (it may return too many ingredients).

```
//ingredient[not(contains(following-sibling::preparation,
                        @name))]/@name
```