Artificial Intelligence 2 – SS 2020
Assignment 6: Return of Bayesian Networks
– Given May 27., Due June 12. –

Hint: Exercises need to be handed in via StudOn at 23:59 on the day they are due or earlier. Please use only the exercise group of your tutor to hand in your work.

If any concepts here seem unfamiliar to you or you have no idea how to proceed, consult the lecture materials, ask a fellow student, your tutor, or on the Forum.

If a problem asks for code, comment it or make it otherwise self-explanatory. You do not need to write a lot, but it should be enough to convince your tutor that you understand what the code does. We may deduct up to 30% for uncommented and unclear code, but would prefer not to.

Problems with no points (0pt) will not be graded, but might appear on the exam in a similar form. For these, we will provide a reference solution after the submission deadline. If you find the reference solution unclear, ask about it on the forum or in in a tutorial.

Problem 6.1 (Bayesian Networks in Python) 100pt

The goal of this exercise is to implement inference by enumeration (or another inference algorithm of your choice) in Bayesian networks in Python. You can find the necessary files at `https://kwarc.info/teaching/AI/resources/bayes.zip`.

Your task is to implement the `query` function in `bayes.py`. Use `test.py` for testing your implementation.

Important: We will test your code automatically. So please make sure that:

- The tests in `test.py` work on your code (without any modifications to `test.py`)
- You use a recent Python version (≥ 3.5)
- You don't use any libraries
- You only upload a single file `bayes.py` with your implementation of `query`

Otherwise you risk getting no points.

The networks are represented as Python dictionaries. Below is a sketch of the network for the burglary example from the lecture notes. It states, for example, that $P(alarm|burglary, \neg earthquak$ 0.94. It follows that $P(\neg alarm|burglary, \neg earthquake) = 1 - 0.94$.

```
{
    'Burglary': ..., 'Earthquake': ...,
    'Alarm': {
        'name': 'Alarm',
        'parents': ['Burglary', 'Earthquake'],
        'probabilities': {
            (True, True): 0.95,
            (True, False): 0.94,
            (False, True): 0.29,
            (False, False): 0.001}
        },
    'JohnCalls': ..., 'MaryCalls': ...
}
```

Tip: The probability tables use tuples as keys. If you prefer to work with lists (recommended, because they are more flexible), you can convert convert them to tuples with the `tuple` function (e.g. `tuple([True, False])` results in `(True, False)`).

Hint: Implement a helper function for getting full joint probabilities.
