Ex 1:

1. $k^N - 1$
2. One probability for each value of each node, so $N * k$ or $4k$
3. The maximum number of parameters is achieved in a network that has 4 nodes in a chain. In this case the first node has k parameters, second k^2, third k^3 and fourth k^4 for a total of $k + k^2 + k^3 + k^4$ parameters.

Ex 2:

```
> python 4.2.py
first_B_share=1.0, second_S_share=0.9900089083807305, third_S_share=0.9889092017062766
```

1. $P(B \mid R, G, \neg S) = 1.0$
2. $P(S \mid R, I, G) \approx 0.99$
3. $P(S \mid \neg R, I, G) \approx 0.99$
4. The answers do make sense. The radio is dependent on the battery in the first one, and in the other two the car starting up is not dependant on the radio working.

Ex 3:

```
> python 4.3.py
share_B_if_A=0.14414893617021277, share_B_if_A_and_E=0.0
```

1. $0.144 \ldots \approx 14.4\ \%$
2. $0.0 = 0\%$
3. $P(B \mid A)$ and $P(B \mid A, E)$
4. They do make sense, but the alarm triggers are hard to combine since there are so few alarms each year.
5. The results do change a bit, and higher n does improve them. $n = 100\ 000$ is probably not quite enough to be perfectly reliable since $1\ 000\ 000$ is still very fast to calculate.

Ex 4:

3. With only 100 steps some of the numbers are very hard to recognise. Usually these are very similar characters such as 4 and 7 or 6 and 9. With max steps the error is usually very small except for these specific characters, where it still ends up at only 30% which is low compared to the 100 step comparisons.