1. Bayesian Inference

a) Write out the equation for calculating joint probabilities, P(Smart,Study,Pass).

P(Smart, Study, Pass) = P(Smart, Study | Pass)*P(Pass)

$$= \frac{P(Pass \mid Smart, Study) * P(Smart, Study)}{P(Pass)} * P(Pass)$$

= **P(Pass | Smart, Study)*P(Smart, Study)**

b) Calculate all the entries in the full joint probability table (JPT) [a 4x2 matrix, like Fig 12.3 in the textbook; [Note: names of variables are capitalized, lower-case indicates truth value, e.g. 'pass' means Pass=T, and '-pass' means Pass=F.]

P(Pass) = 0.492

	smart		-smart	
	study	-study	study	-study
pass	.95*.3*.4=0.114	0.126	0.168	0.084
¬pass	0.006	0.054	0.112	0.336

c) From the JPT, compute the probability that a student is smart, given that they pass the test but did not study.

P(smart | pass, ¬study) = P(smart, pass, -study) / P(pass, -study)

= 0.126/(0.126 + 0.084) = **0.6**

d) From the JPT, compute the probability that a student did not study, given that they are smart but did not pass the test.

P(¬study | smart, ¬pass) = P(¬study, smart, -pass) / P(smart, ¬pass)

= 0.054 / (0.006 + 0.054) = **0.9**

e) Compute the marginal probability that a student will pass the test given that they are smart

P(pass | smart) = 0.114 + 0.126 = **0.24**

f) Compute the marginal probability that a student will pass the test given that they study

P(pass | study) = 0.114 + 0.168 = **0.282**

2. Bayesian Networks

a) Using Equation 13.2 in the textbook (p. 415), write out the expression for the joint probability for any state (i.e. combination of truth values for the 5 variables in this problem). [Note: Use capital letters for names of variables, and lower-case to indicate truth value, e.g. 'cold' means Cold=T, and '-cold' means Cold=F.]

P(Cold,Sneeze,Allergic,Scratches,Cat) = **P(Scratches | Cat)*P(Allergy | Cat)*P(Sneeze | Allergy, Cold)*P(Cold)*P(Cat)**

b) Use the equation above to calculate the joint probability that the person sneezes, but does not have a cold, has a cat, is allergic, and there are scratches on the furniture:
P(-cold,sneeze,allergic,scratches,cat) = P(scratches | cat)*P(allergy | cat)*P(sneeze | allergy, -cold)*P(-cold)*p(cat) = 0.5*0.75*0.7*0.95*0.02 = **0.00499**

c) Use normalization to calculate the conditional probability that a person has cat, given that they sneeze and are allergic to cats, but do not have a cold, and there are scratches on the furniture.

P(cat|-cold,sneeze,allergic,scratches) = P(-cold,sneeze,allergic,scratches,cat) / P(-cold,sneeze,allergic,scratches)
= 0.00499 /
P(-cold)*P(sneeze | allergy, -cold)*P(allergic)*P(scratches)
= 0.00499 / (0.95*0.7*(0.05+0.75)*(0.5+0.05))
= **0.0171**

d) Use Bayes' Rule to re-write the expression for P(cat|scratches). Look up the values for the numerator in the table above.
P(cat|scratches) = **P(scratches | cat)*P(cat) / P(scratches)**

e) The denominator in the answer for (d) would require marginalization over how many joint probabilities? Write out the expressions for these (i.e. expand the denominator, but you don't have to calculate the actual values).
2
P(scratches) = P(-cat, scratches) + P(cat, scratches)

3. PDDL and Situation Calculus

To start a car, you have to be at the car and have the key, and the car has to have a charged battery and the tank has to have gas. Afterwards, the car will be running, and you will still be at the car and have the key after starting the engine.

a. Write a PDDL operator to describe this action.

startCar(car, key, battery, gasTank):
  pre-conds: (at(car), hasKey(key), charged(battery), hasGas(gasTank))
  effects: (running(car), at(car), hasKey(key), charged(battery), hasGas(gasTank))

b. Describe the same operator using Situation Calculus (remember to add a situation argument to your predicates).

∀s, car, key, battery, gasTank, s_next :
(at(car, s) ∧ hasKey(key, s) ∧ charged(battery, s) ∧ hasGas(gasTank, s) →
    (running(car, s) ∧
    at(car, s) ∧
    hasKey(key, s_next) ∧
    charged(battery, s_next) ∧
    hasGas(gasTank, s_next) ∧
    situationNext(s, s_next) ∧
    do(startCar(car, key, battery, gasTank), s) = s_next)

c. Add a Frame Axiom that says that starting this car will not change whether any other car is out of gas (tank empty).

∀s, s_next, car, other_car :
  (car ≠ other_car) → (outOfGas(other_car, s) ⇔ outOfGas(other_car, s_next))