PART A)
Code:

```
import random
from numpy import *

# PART A)
print("PART A)")

# going to be using this method for all parts of the problem:
def compute_probability(sky_var, sprinkler_var, rain_var, grass_var):
    sky_p = .5 # P(sky=sunny) = P(sky=cloudy) = 5, so we can just intialize sky_p to .5
    # initializing the rest of the probabilities:
    sprinkler_p = 0
    rain_p = 0
    grass_p = 0

    # the following conditionals are derived from the Bayesian network given in the assignment:
    if sky_var == "cloudy":
        if sprinkler_var == "on":
            sprinkler_p = .5
        elif sprinkler_var == "off":
            sprinkler_p = 1 - .5
        else:
            raise "ERROR: UNDEFINED VARIABLE"
        if rain_var == "yes":
            rain_p = .7
        elif rain_var == "no":
            rain_p = 1 - .7
        else:
            raise "ERROR: UNDEFINED VARIABLE"
    elif sky_var == "sunny":
        if sprinkler_var == "on":
            sprinkler_p = .9
        elif sprinkler_var == "off":
            sprinkler_p = 1 - .9
        else:
            raise "ERROR: UNDEFINED VARIABLE"
        if rain_var == "yes":
            rain_p = .01
        elif rain_var == "no":
            rain_p = 1 - .01
        else:
            raise "ERROR: UNDEFINED VARIABLE"
    else:
        raise "ERROR: UNDEFINED VARIABLE"
```

```
    if sprinkler_var == "on" and rain_var == "yes":
        if grass_var == "dry":
            grass_p = 1 - .99
        elif grass_var == "wet":
            grass_p = .99
        else:
            raise "ERROR: UNDEFINED VARIABLE"
    # combining these two conditionals because they have the same result:
    elif (sprinkler_var == "on" and rain_var == "no") or (sprinkler_var == "off" and rain_var == "yes"):
        if grass_var == "dry":
            grass_p = 1 - .9
        elif grass_var == "wet":
            grass_p = .9
        else:
            raise "ERROR: UNDEFINED VARIABLE"
    elif sprinkler_var == "off" and rain_var == "no":
        if grass_var == "dry":
            grass_p = 1 - .01
        elif grass_var == "wet":
            grass_p = .01
        else:
            raise "ERROR: UNDEFINED VARIABLE"
    else:
        raise "ERROR: UNDEFINED VARIABLE"

    return sky_p * sprinkler_p * rain_p * grass_p

# the domains of the variables:
sky = ["cloudy", "sunny"]
sprinkler = ["on", "off"]
rain = ["yes", "no"]
grass = ["dry", "wet"]
# array for the 2^4 probabilities:
probabilities = []

# the following intializations are for PART B):
grass_dry_probabilities = []
grass_wet_probabilities = []

for i in sky:
    for j in sprinkler:
        for k in rain:
            for l in grass:
                p = compute_probability(i, j, k, l)
                p_sig = '%s' % float('%.5g' % p) # reduce significant digits
                print("P(sky=" + i + ", sprinkler=" + j + ", rain=" + k + ", grass=" + l + ") =", p_sig)
                probabilities.append(p)
                # the following code is for PART B):
                if l == "dry":
                    grass_dry_probabilities.append(p)
                if l == "wet":
                    grass_wet_probabilities.append(p)

print("sum of probabilities:", sum(probabilities))
print("\n")
```

Output:

```
PART A)
P(sky=cloudy, sprinkler=on, rain=yes, grass=dry) = 0.00175
P(sky=cloudy, sprinkler=on, rain=yes, grass=wet) = 0.17325
P(sky=cloudy, sprinkler=on, rain=no, grass=dry) = 0.0075
P(sky=cloudy, sprinkler=on, rain=no, grass=wet) = 0.0675
P(sky=cloudy, sprinkler=off, rain=yes, grass=dry) = 0.0175
P(sky=cloudy, sprinkler=off, rain=yes, grass=wet) = 0.1575
P(sky=cloudy, sprinkler=off, rain=no, grass=dry) = 0.07425
P(sky=cloudy, sprinkler=off, rain=no, grass=wet) = 0.00075
P(sky=sunny, sprinkler=on, rain=yes, grass=dry) = 4.5e-05
P(sky=sunny, sprinkler=on, rain=yes, grass=wet) = 0.004455
P(sky=sunny, sprinkler=on, rain=no, grass=dry) = 0.04455
P(sky=sunny, sprinkler=on, rain=no, grass=wet) = 0.40095
P(sky=sunny, sprinkler=off, rain=yes, grass=dry) = 5e-05
P(sky=sunny, sprinkler=off, rain=yes, grass=wet) = 0.00045
P(sky=sunny, sprinkler=off, rain=no, grass=dry) = 0.049005
P(sky=sunny, sprinkler=off, rain=no, grass=wet) = 0.000495
sum of probabilities: 1.0
```

For PART A), we needed to compute the probability of all given scenarios in the given Bayesian Network. I wrote a method called “compute_probability”, which contains all the probabilities (both prior and conditional) from the Bayesian Network. It computes 4 probabilities (one for every variable), and then multiplies them together to get the probability that all 4 are the case. The figure above demonstrates this. There are 4 variables, each with a domain of size 2, so there are 2^4 = 16 possible scenarios. There are no other possible scenarios, so all of the probabilities add up to 1.

PART B)
Code:

```
# PART B)
print("PART B)")

P_grass_dry = sum(grass_dry_probabilities)
P_grass_wet = sum(grass_wet_probabilities)
# these are to confirm that each table sums to 1:
grass_dry_sum = 0
grass_wet_sum = 0

# for dry grass:
for i in sky:
    for j in sprinkler:
        for k in rain:
            p = compute_probability(i, j, k, "dry") / P_grass_dry
            # not doing significant digits here because decimal values are not as nice as in PART A):
            print("P(sky=" + i + ", sprinkler=" + j + ", rain=" + k , "| grass=dry) =", p)
            grass_dry_sum = grass_dry_sum + p

# the way python adds it comes out to .999999, so rounding to exactly 1:
print("sum of probabilities =", '%s' % float('%.5g' % grass_dry_sum))
print();

# for wet grass:
for i in sky:
    for j in sprinkler:
        for k in rain:
            p = compute_probability(i, j, k, "wet") / P_grass_wet
            print("P(sky=" + i + ", sprinkler=" + j + ", rain=" + k , "| grass=wet) =", p)
            grass_wet_sum = grass_wet_sum + p

print("sum of probabilities =", grass_wet_sum)
print("\n")
```

Output:

```
PART B)
P(sky=cloudy, sprinkler=on, rain=yes | grass=dry) = 0.008990495761623434
P(sky=cloudy, sprinkler=on, rain=no | grass=dry) = 0.038530696121243255
P(sky=cloudy, sprinkler=off, rain=yes | grass=dry) = 0.08990495761623424
P(sky=cloudy, sprinkler=off, rain=no | grass=dry) = 0.3814538916003083
P(sky=sunny, sprinkler=on, rain=yes | grass=dry) = 0.0002311841767274598
P(sky=sunny, sprinkler=on, rain=no | grass=dry) = 0.22887233496018491
P(sky=sunny, sprinkler=off, rain=yes | grass=dry) = 0.000256871307474955
P(sky=sunny, sprinkler=off, rain=no | grass=dry) = 0.2517595684562034
sum of probabilities = 1.0

P(sky=cloudy, sprinkler=on, rain=yes | grass=wet) = 0.21512385919165578
P(sky=cloudy, sprinkler=on, rain=no | grass=wet) = 0.08381449059415164
P(sky=cloudy, sprinkler=off, rain=yes | grass=wet) = 0.19556714471968709
P(sky=cloudy, sprinkler=off, rain=no | grass=wet) = 0.0009312721177127958
P(sky=sunny, sprinkler=on, rain=yes | grass=wet) = 0.005531756379214007
P(sky=sunny, sprinkler=on, rain=no | grass=wet) = 0.4978580741292606
P(sky=sunny, sprinkler=off, rain=yes | grass=wet) = 0.0005587632706276773
P(sky=sunny, sprinkler=off, rain=no | grass=wet) = 0.000614639597690445
sum of probabilities = 1.0
```

For PART B), we needed to compute the probability of all given scenarios when the grass is dry. Then we needed to do the same thing for when the grass is wet. I used this formula:

P(sky, sprinkler, rain | grass=dry) = P(sky, sprinkler, rain, grass=dry) / P(grass=dry)

I did the same thing for wet grass. This gave me 2 tables of probabilities. Because there are 3 variables other than grass, each table is 2^3 = 8 rows. Both tables also add to 1, because both tables are normalized. In other words, each table assumes that the grass is in a certain state, and then enumerates each scenario given that assumption.

PART C)
Code:

```
# PART C)
print("PART C)")
samples = 1000

# for dry grass:
dry_sky_samples = []
dry_sprinkler_samples = []
dry_rain_samples = []

# sample 0 (picked randomly):
dry_sky_samples.append("cloudy")
dry_sprinkler_samples.append("on")
dry_rain_samples.append("yes")

for i in range(1, samples):
    # need to derive the probability from the previous samples
    # grass is always dry for this sampling
    sky_sample_p = compute_probability("cloudy", dry_sprinkler_samples[i-1], dry_rain_samples[i-1], "dry") / (compute_probability("sunny", dry_sprinkler_samples[i-1],
dry_rain_samples[i-1], "dry") + compute_probability("cloudy", dry_sprinkler_samples[i-1], dry_rain_samples[i-1], "dry"))
    sprinkler_sample_p = compute_probability(dry_sky_samples[i-1], "on", dry_rain_samples[i-1], "dry") / (compute_probability(dry_sky_samples[i-1], "off", dry_rain_samples[i-1],
"dry") + compute_probability(dry_sky_samples[i-1], "on", dry_rain_samples[i-1], "dry"))
    rain_sample_p = compute_probability(dry_sky_samples[i-1], dry_sprinkler_samples[i-1], "yes", "dry") / (compute_probability(dry_sky_samples[i-1], dry_sprinkler_samples[i-1],
"no", "dry") + compute_probability(dry_sky_samples[i-1], dry_sprinkler_samples[i-1], "yes", "dry"))

    # picking a random number; if rand <= p, add the corresponding condition to the samples array
    rand = random.uniform(0, 1)
    if rand <= sky_sample_p:
        dry_sky_samples.append("cloudy")
    else:
        dry_sky_samples.append("sunny")

    rand = random.uniform(0, 1)
    if rand <= sprinkler_sample_p:
        dry_sprinkler_samples.append("on")
    else:
        dry_sprinkler_samples.append("off")

    rand = random.uniform(0, 1)
    if rand <= rain_sample_p:
        dry_rain_samples.append("yes")
    else:
        dry_rain_samples.append("no")

sum_ = 0
index = 0
for i in sky:
    for j in sprinkler:
        for k in rain:
            count = 0
            for l in range(samples):
                # this finds each instance of a scenario and counts it:
                if dry_sky_samples[l] == i and dry_sprinkler_samples[l] == j and dry_rain_samples[l] == k:
                    count = count + 1
            percent_error = 100*abs(count/samples - grass_dry_probabilities[index])
            pe_sig = '%s' % float('%.2g' % percent_error)
            # displays the percent of samples that were the given scenario:
            print("gibbs estimated probability(sky=" + i + ", sprinkler=" + j + ", rain=" + k , "| grass=dry) =", count/samples, "-> percent error =", pe_sig)
            index = index + 1
            sum_ = sum_ + count/samples

print()

# for wet grass:
wet_sky_samples = []
wet_sprinkler_samples = []
wet_rain_samples = []
wet_sky_samples.append("cloudy")
wet_sprinkler_samples.append("on")
wet_rain_samples.append("yes")

for i in range(1, samples):
    sky_sample_p = compute_probability("cloudy", wet_sprinkler_samples[i-1], wet_rain_samples[i-1], "wet") / (compute_probability("sunny", wet_sprinkler_samples[i-1],
wet_rain_samples[i-1], "wet") + compute_probability("cloudy", wet_sprinkler_samples[i-1], wet_rain_samples[i-1], "wet"))
    sprinkler_sample_p = compute_probability(wet_sky_samples[i-1], "on", wet_rain_samples[i-1], "wet") / (compute_probability(wet_sky_samples[i-1], "off", wet_rain_samples[i-1],
"wet") + compute_probability(wet_sky_samples[i-1], "on", wet_rain_samples[i-1], "wet"))
    rain_sample_p = compute_probability(wet_sky_samples[i-1], wet_sprinkler_samples[i-1], "yes", "wet") / (compute_probability(wet_sky_samples[i-1], wet_sprinkler_samples[i-1],
"no", "wet") + compute_probability(wet_sky_samples[i-1], wet_sprinkler_samples[i-1], "yes", "wet"))

    if rand <= sky_sample_p:
        wet_sky_samples.append("cloudy")
    else:
        wet_sky_samples.append("sunny")
    rand = random.uniform(0, 1)
    if rand <= sprinkler_sample_p:
        wet_sprinkler_samples.append("on")
    else:
        wet_sprinkler_samples.append("off")
    rand = random.uniform(0, 1)
    if rand <= rain_sample_p:
        wet_rain_samples.append("yes")
    else:
        wet_rain_samples.append("no")

sum_ = 0
index = 0
for i in sky:
    for j in sprinkler:
        for k in rain:
            count = 0
            for l in range(samples):
                if wet_sky_samples[l] == i and wet_sprinkler_samples[l] == j and wet_rain_samples[l] == k:
                    count = count + 1
            percent_error = 100*abs(count/samples - grass_wet_probabilities[index])
            pe_sig = '%s' % float('%.2g' % percent_error)
            print("gibbs estimated probability(sky=" + i + ", sprinkler=" + j + ", rain=" + k , "| grass=wet) =", count/samples, "-> percent error =", pe_sig)
            index = index + 1
            sum_ = sum_ + count/samples

print("\n")
```

Output:

```
PART C)
gibbs_estimated_probability(sky=cloudy, sprinkler=on, rain=yes | grass=dry) = 0.009 -> percent error = 0.00095
gibbs_estimated_probability(sky=cloudy, sprinkler=on, rain=no | grass=dry) = 0.137 -> percent error = 9.8
gibbs_estimated_probability(sky=cloudy, sprinkler=off, rain=yes | grass=dry) = 0.052 -> percent error = 3.8
gibbs_estimated_probability(sky=cloudy, sprinkler=off, rain=no | grass=dry) = 0.351 -> percent error = 3.0
gibbs_estimated_probability(sky=sunny, sprinkler=on, rain=yes | grass=dry) = 0.001 -> percent error = 0.077
gibbs_estimated_probability(sky=sunny, sprinkler=on, rain=no | grass=dry) = 0.105 -> percent error = 12.0
gibbs_estimated_probability(sky=sunny, sprinkler=off, rain=yes | grass=dry) = 0.043 -> percent error = 4.3
gibbs_estimated_probability(sky=sunny, sprinkler=off, rain=no | grass=dry) = 0.302 -> percent error = 5.0

gibbs_estimated_probability(sky=cloudy, sprinkler=on, rain=yes | grass=wet) = 0.119 -> percent error = 9.6
gibbs_estimated_probability(sky=cloudy, sprinkler=on, rain=no | grass=wet) = 0.224 -> percent error = 14.0
gibbs_estimated_probability(sky=cloudy, sprinkler=off, rain=yes | grass=wet) = 0.078 -> percent error = 12.0
gibbs_estimated_probability(sky=cloudy, sprinkler=off, rain=no | grass=wet) = 0.035 -> percent error = 3.4
gibbs_estimated_probability(sky=sunny, sprinkler=on, rain=yes | grass=wet) = 0.179 -> percent error = 17.0
gibbs_estimated_probability(sky=sunny, sprinkler=on, rain=no | grass=wet) = 0.362 -> percent error = 14.0
gibbs_estimated_probability(sky=sunny, sprinkler=off, rain=yes | grass=wet) = 0.001 -> percent error = 0.044
gibbs_estimated_probability(sky=sunny, sprinkler=off, rain=no | grass=wet) = 0.002 -> percent error = 0.14
```

For PART C), I had to implement Gibbs Sampling to estimate the probability of the scenarios from PART B). I started with a random sample, then in a for loop I created each new sample by drawing on the previous sample and a Bernoulli random number. This gave me numbers that were similar to the values from PART B). I also calculated the percent error by finding the percent difference between these results and the results in PART B). There was a little more error than I expected; my Gibbs sampling might have been slightly different.

PART D)
Code:

```
# PART D)
print("PART D)")

# I already coded it with strings (cloudy, on, etc.) before this part. I found it easier to complete the assignment using strings,
# so I am leaving the code above as is and converting into 1s and 0s just for this part:
for i in range(samples):
    if dry_sky_samples[i] == "cloudy":
        dry_sky_samples[i] = 0
    if dry_sky_samples[i] == "sunny":
        dry_sky_samples[i] = 1

    if dry_sprinkler_samples[i] == "on":
        dry_sprinkler_samples[i] = 0
    if dry_sprinkler_samples[i] == "off":
        dry_sprinkler_samples[i] = 1

    if dry_rain_samples[i] == "yes":
        dry_rain_samples[i] = 0
    if dry_rain_samples[i] == "no":
        dry_rain_samples[i] = 1

    if wet_sky_samples[i] == "cloudy":
        wet_sky_samples[i] = 0
    if wet_sky_samples[i] == "sunny":
        wet_sky_samples[i] = 1

    if wet_sprinkler_samples[i] == "on":
        wet_sprinkler_samples[i] = 0
    if wet_sprinkler_samples[i] == "off":
        wet_sprinkler_samples[i] = 1

    if wet_rain_samples[i] == "yes":
        wet_rain_samples[i] = 0
    if wet_rain_samples[i] == "no":
        wet_rain_samples[i] = 1

# for dry grass:
dry_samples = [dry_sky_samples, dry_sprinkler_samples, dry_rain_samples]
dry_samples_matrix = corrcoef(dry_samples)
print("Pairwise Correlation Between Gibbs Samples For Dry Grass")
print(dry_samples_matrix)
print()

# for wet grass:
wet_samples = [wet_sky_samples, wet_sprinkler_samples, wet_rain_samples]
wet_samples_matrix = corrcoef(wet_samples)
print("Pairwise Correlation Between Gibbs Samples For Wet Grass")
print(dry_samples_matrix)
```

Output:

```
PART D)
Pairwise Correlation Between Gibbs Samples For Dry Grass
[[ 1.          0.03542007  0.02199439]
 [ 0.03542007  1.         -0.12367234]
 [ 0.02199439 -0.12367234  1.        ]]

Pairwise Correlation Between Gibbs Samples For Wet Grass
[[ 1.          0.03542007  0.02199439]
 [ 0.03542007  1.         -0.12367234]
 [ 0.02199439 -0.12367234  1.        ]]
```

For PART D), we needed to compute the pairwise correlation between the samples for both dry and wet grass. I did this using the corrcoef function from python's library. This returns Pearson product-moment correlation coefficients. One can notice that there is a negative correlation in the matrices. Furthermore, this is the normalized covariance matrix, so there are 1s running along the diagonal of each matrix.

PART E)
Code:
NOTE: These pieces of code are not consecutive- for this part, I only changed two things (probabilities in the “compute_probability” function and the sample size).

```
# the following conditionals are derived from the Bayesian network given in the assignment:
if sky_var == "cloudy":
    if sprinkler_var == "on":
        sprinkler_p = .1
    elif sprinkler_var == "off":
        sprinkler_p = 1 - .1
    else:
        raise "ERROR: UNDEFINED VARIABLE"
    if rain_var == "yes":
        rain_p = .9
    elif rain_var == "no":
        rain_p = 1 - .9
    else:
        raise "ERROR: UNDEFINED VARIABLE"
elif sky_var == "sunny":
    if sprinkler_var == "on":
        sprinkler_p = .9
    elif sprinkler_var == "off":
        sprinkler_p = 1 - .9
    else:
        raise "ERROR: UNDEFINED VARIABLE"
    if rain_var == "yes":
        rain_p = .01
    elif rain_var == "no":
        rain_p = 1 - .01
    else:
        raise "ERROR: UNDEFINED VARIABLE"
else:
    raise "ERROR: UNDEFINED VARIABLE"


if sprinkler_var == "on" and rain_var == "yes":
    if grass_var == "dry":
        grass_p = 1 - .99
    elif grass_var == "wet":
        grass_p = .99
    else:
        raise "ERROR: UNDEFINED VARIABLE"
# combining these two conditionals because they have the same result:
elif (sprinkler_var == "on" and rain_var == "no") or (sprinkler_var == "off" and rain_var == "yes"):
    if grass_var == "dry":
        grass_p = 1 - .9
    elif grass_var == "wet":
        grass_p = .9
    else:
        raise "ERROR: UNDEFINED VARIABLE"
elif sprinkler_var == "off" and rain_var == "no":
    if grass_var == "dry":
        grass_p = 1 - .01
    elif grass_var == "wet":
        grass_p = .01
    else:
        raise "ERROR: UNDEFINED VARIABLE"
else:
    raise "ERROR: UNDEFINED VARIABLE"
```

```
# PART C)
print("PART C)")
samples = 100
```

Output:

```
PART E)
Pairwise Correlation Between Gibbs Samples For Dry Grass
[[ 1.         -0.05327728  0.06543611]
 [-0.05327728  1.         -0.29692421]
 [ 0.06543611 -0.29692421  1.        ]]

Pairwise Correlation Between Gibbs Samples For Wet Grass
[[ 1.         -0.05327728  0.06543611]
 [-0.05327728  1.         -0.29692421]
 [ 0.06543611 -0.29692421  1.        ]]
```

As the relationship becomes stronger (the probabilities are closer to 1 or 0), there is more error. This is because it is harder to get rare outcomes. Take the binary variable sky for example (it either has a value of 0 or 1). It is a property of Gibbs Sampling that the samples are nearby each other, so there will mostly be samples drawn at the edges (.99 or .01 or whatever it is). For instance, if there are 1000 ideally, ideally you'd want 990 at 0 and 10 at 1 (if the probability is distribution is such that P(0) = .99 and P(1) = .01), but there is more likely going to be ~999 or so at 0, and ~1 at 1. This is all exaggerated by the smaller amount of samples.