Andrew Yang
V00878595

Assignment 4

Part 1:

1.
```
from kanren import Relation, facts

def main():
    parent = Relation()
    facts(parent, ("Darth Vader", "Luke Skywalker"),
                  ("Darth Vader", "Leia Organa"),
                  ("Leia Organa", "Kylo Ren"),
                  ("Han Solo", "Kylo Ren"))
    # Darth Vader is Luke and Leia's Parent
    # Leia and Han are Kylo's Parents
```

2.
```
run(1, x, parent(x, "Luke Skywalker"))
# What is x, such that x is the parent of Luke
run(2, y, parent("Darth Vader", y))
# What is y, such that y is the children of Vader
```

3.
```
    grandparent = Relation()
    facts(grandparent, ("Darth Vader", "Kylo Ren"))
    # Darth Vader is Kylo Ren's Grandparent

    run(1, z, grandparent(z, "Kylo Ren"))
    # What is z, such that z is the grandparent of Kylo Ren
    #OR
    y = var()
    run(1, x, parent(x, y),
              parent(y, "Kylo Ren"))
    # What is x, such that x is the parent of y, where y is the parent
    # of Kylo Ren
```

4.

```
def logicPy():
    # Create a separate dictionary for each character's tree
    Luke = {
        "Parent": "Darth Vader",
        "Sister": "Leia Organa",
        "Brother-in-Law": "Han Solo",
        "Nephew": "Kylo Ren"
    }

    Leia = {
        "Parent": "Darth Vader",
        "Brother": "Luke Skywalker",
        "Spouse": "Han Solo",
        "Son": "Kylo Ren"
    }

    Vader = {
        "Son": "Luke Skywalker",
        "Daughter": "Leia Organa",
        "Grandson": "Kylo Ren",
        "Son-in-Law": "Han Solo"
    }

    Kylo = {
        "Mother": "Leia Organa",
        "Father": "Han Solo",
        "Grandfather": "Darth Vader",
        "Uncle": "Luke Skywalker"
    }

    print(Luke["Parent"])
    # Print the parent of Luke Skywalker
    print(Vader["Son"] + Vader["Daughter"])
    # Print the son and daughter of Darth Vader
    print(Kylo["Grandparent"])
    # Print the grandfather of Kylo Ren
```

Looking at the two styles of programming this question, either with logpy or dictionaries, it's pretty clear both methods are pretty straight forward and easy to implement. However, we only had five nodes to consider in this case. When we start having larger datasets, it becomes quite difficult to manage. Using, logpy, we're able to establish relations, which can function as bi-directional edges between two nodes, or family members. Using dictionaries, we can see that if we were to create tables for every family member, we require exponential space to create the relations.

Part 2:
1 & 2.

```
#Andrew Yang
#V00878595

import matplotlib.pyplot as plt
from scipy import stats
import numpy as np
from hmmlearn import hmm

def main():
    model_states = hmm.MultinomialHMM(n_components=2)
    model_states.startprob_ = np.array([1.0, 0.0])

    model_states.transmat_ = np.array([[0.7, 0.3],
                                       [0.5, 0.5]])

    model_states.emissionprob_ = np.array([[0.2, 0.1, 0.7],
                                           [0.3, 0.6, 0.1]])

    X, Z = model_states.sample(300)
```

3 & 4.

```
    remodel = hmm.MultinomialHMM(n_components=2, n_iter=300)
    remodel.fit(X)

    predict = remodel.predict(X)

    print("Learned transition matrix")
    print(remodel.transmat_)
    print("Learned emission probabilities")
    print(remodel.emissionprob_)

    print(predict)
```

For question 3, these are the learned transition matrix and emission probabilities

```
Learned transition matrix
[[0.39891735 0.60108265]
 [0.43987555 0.56012445]]
Learned emission probabilities
[[0.40711919 0.10275128 0.49012953]
 [0.03571642 0.41067935 0.55360423]]
```

For reference, these are the original transition probabilities

	Healthy	Injured
Healthy	0.7	0.3
Injured	0.5	0.5

	Dribble	Pass	Shoot
Healthy	0.2	0.1	0.7
Injured	0.3	0.6	0.1

As you can see from the numbers, they're off quite a bit from the learned transition and emission matrices. The transition from injured to each state came close to being 0.5/0.5 split. However, the transitions from healthy to each state was off by .3.

The emission probabilities had some variables like P(healthy|pass) that was close to the original, but the other transition probabilities are off by quite a bit as well. As noted by my classmates, this is "caused by doing unsupervised learning of the hidden parameters with random initialization values so you can get very different results run after run."

```
Original model:
[0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 0 0
 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1
 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0
 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 1 1 1 1
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1
 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0
 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 0 1 0 1 0
 0 0 0 0]

Learned model:
[0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1
 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1
 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0
 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0
 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0
 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1
 1 1 1 1]
```

The two models differ by about 185 states. I could've counted wrong, but it was around that value. Out of 300 observations that means the predicted model had more than 50% error if we were using the original model as the ground truth.

*Note: I didn't have the original model printed out the first time I ran it, so the values from this model might differ from the transition matrices a bit. These are the transition matrices for the sequences generated.

```
Learned transition matrix
[[0.65519299 0.34480701]
 [0.31375436 0.68624564]]
Learned emission probabilities
[[0.47675905 0.47032884 0.05291211]
 [0.03440689 0.15568863 0.80990449]]
```