**Intro**:

A Bayesian network is a data structure that stores the probabilities of a number of events through individual nodes. Two nodes will be connected in a parent child relationship only if they are probabilistically dependent on each other. The alternative would be to store the complete joint probability table that would be exponential in time and space complexity. A Bayesian network solves this problem by utilizing the concept of conditional independence, and thus it only need to look at the relevant nodes to the probabilistic question at hand.

Expectation maximization is a general methodology that can be used within Bayesian networks to find good estimate probabilities even given incomplete datasets. Overall it works by using a model to fill in the dataset with probability distributions and then it learns a new model based on this filled in version of the dataset. Through repetition this process will converge at a local maxima and so it can be halted when the difference between the likelihood of a model at step n+1 is not significantly different than the likelihood of the model generated at step n.

The problem at hand is figuring out the correlation between the three variables: gender, height, and weight from datasets where some of the information regarding gender is missing. The relationship is one where gender is assumed to be the only parent of both height and weight and each variable is assumed to be a binary value of 0 or 1.

**Methods**:

I approached this problem by creating a general solution to any case of missing data within any Bayesian network. Although this solution took me longer than the allotted time I can easily adapt my program to any similar relaxed problem such as where the network is composed of many more nodes and various types of parent child relationships, as well as where other and multiple nodes are missing data. I designed two classes and a driver script in python. One class represents a single node in the network and can add events relative to its parents and use that to calculate and store probabilities. The other class represents the collections of nodes as a network and can read in a dataset into a network, calculate joint probabilities, and can perform expectation maximization on such a dataset if needed.

The algorithm works by performing the following general steps. It generates a modified dataset without any missing data by dividing each line of the dataset that was missing information into multiple lines, each new line representing a potential value of the missing data. It assigns a probability that each line occurs based on the value assigned either by using initial data which is hard coded in or by using random valid distributions using the range of values that could be there. It then uses that new dataset to generate the conditional probabilities of the network. Lastly, it takes the original dataset with missing data and repeats the first step with the new learned probabilities achieved in the second step. After each two steps the probability of the original dataset given the current model is generated by calculating the log likelihood of the model. This step requires that any missing data is ignored and the rest of the data is used to find the joint probability of the remaining information. After each calculation of likelihood of the current iterations is compared to the likelihood of the previous one. If the change is an improvement large of for consideration then the algorithm is run again I also created a driver function that would run the expectation maximization algorithm multiple times and compare the final resulting values such that the best network could saved. All of this was run 2x on each of five files, one time working with the set initial values, and one time using random valus.

**Results**:

Since my overall algorithm ran through various iterations where each of those iterations conducted multiple iterations of expectation maximization with new starting values, I will use the best final values of each outer iteration as I display my data. Through the following plots I plan to show the following comparisons.

1. Plot of iteration vs. likelihood for the best solutions
2. Number of iterations required compared to the amount of missing data
3. Number of iterations required with missing data of a random starting point compared to the given starting point
4. Comparison of non best solutions to best solution

Charts of the details of the best iterations from the program are included at the end. They include the final conditional probability tables.

**Discussion**:

It is very clear from the charts that the more missing data leads the more iterations will occur. Additionally, it is possible to see the convergence of every run o the algorithm. Also, when comparing the number of iterations a random start required vs. the given start values it is possible to see that the given start values never required as many iterations. This means the start values were close to a local (or using meta-logic an optimal) maxima.

Based on the way that my program output the comparison between different starting points there is not a clear graph to express the difference. Still by counting the number of times in the random starting point files that it states that it found a better answer you can see that will minimal missing data multiple starting points will only lead you to revise your answer 1 or maybe 2x out of 9 iterations, but with 100% missing data the program revises its answer 3x. This means that the more missing data there is the more important it is to test multiple starting points since the algorithm is very dependent on its starting points and the more missing data the greater chance the algorithm will get stuck at a far from optimal local maxima.

Except for the case of missing 50% of the gender data the likelihood of the best random starting points and the given starting points tend to be fairly close and off only by the thousandths place or less. This suggests the effectiveness of the algorithm in most scenarios as random start points are close to known start point.

While it is clear that the amount of missing data has an effect on the way in which the algorithm runs, how many iterations it requires, and how often it will find a better answer using different starting points it seems that it still provides a sensible answer given what it knows. What is interesting is that the likelihood of the models with more missing data is greater than of the ones with less missing data and this can be explained by stating that since there is more missing data there is less data that occurred and therefore more ways it may have occurred making any answer more likely to have occurred. An interesting test to see the effect of missing data would be to create a model from one dataset and see how likely it is for the other dataset. This would truly let you compare the likelihood of a model generated with less data to a model generated with complete or only somewhat missing date.