a. Feature models are usually represented as a tree because this provides the most flexibility for specifying dependencies between different features. For each of the alternatives presented in the assignment, we will list a disadvantage of using that alternative.

- List: the typical representation of a list in CS is a long line of connected nodes. This makes it possible to specify direct dependencies (ie. A depends on B which in turn depends on C). It is possible to overcome this limitation by allowing multiple leaf nodes, thus making it possible to define seperate features. Doing this, however, would make sort of tree from the list.
- Graph: a graph can be used to represent a feature model. This graph would have three special properties:

  1) it does not have cycles (since a feature cannot depend on itself or a feature up its path)
  2) it is connected (otherwise a "feature" would not belong to the product)
  3) it is directed (otherwise we cannot determine the dependance of features)

  This kind of graph actually is a tree.
- Expression: a feature model can be represented as a expression (see for example http://www.cs.utexas.edu/~schwartz/ATS/fopdocs/guidsl.html). However, these kinds of expressions require more intimate understanding of the subject matter (and the language used). A feature tree can more easily be presented to a Product Manager / other executive. Tree's present a visual alternative which they can understand.
- Prolog program: Prolog is used for representing a feature model (just like the expression mentioned above). However, as for the expression: a prolog program requires more knowledge of the language used, which makes it hard to present it to a Product Manager / other executive. Tree's present a visual alternative which they can understand.

b. Due to the possibility of adding constraints in our feature models, we can create two different feature modals that, due to their constraints, result in exactly the same feature selections and products.

   For example, since we have the possibility of a "NOT" and an "OR" construct, we can also create a "NOR" construction by concatenation. This allow us to write "NOT Feature1" as "Feature1 NOR Feature1".

   Although the feature models would be different, the possible choices and products would be exactly the same.

   Possibilities for normalization would include checking for a "concatenated" construction since it is not possible to create different feature models for the same product line without such a concetenated construction. These "concatenated" constructions would include "NAND" and "NOR" like constructions, since these allow us to rewrite constraints in a different way.