1. (12 points). Consider the “object-oriented” implementation of the Core interpreter with classes such as Stmt, SS, corresponding, respectively, to <stmt seq>, <stmt>, etc. For this problem, you have to show how you can use polymorphism in the implementation of the Stmt class, with the classes such as Assign, If, corresponding to the various alternative types of statements being derived classes of the Stmt class. Each of the methods, Parse, Print, and Execute, of Stmt will be an abstract (or, in C++ terminology, pure virtual) method.

Hint: You will have to modify the SS class, corresponding to <stmt seq>, since it is the Parse method of that class which will construct a Stmt object. But, in fact, you cannot construct such an object given that there are abstract methods in the Stmt class. Thinking about how to address this problem will help you answer the question. (The Print and Execute methods should be straightforward.)

2. (8 points). Some people carry polymorphism too far! Specifically, in the example of the Core interpreter, you might have a base class called Node that corresponds to all kinds of nodes that might appear in the abstract parse tree and have classes such as Prog, Decl, Stmt, etc. be derived classes of Node. Node will contain three abstract methods, Parse, Print, Execute, with the derived classes providing the definitions of these methods.

It turns out this is a bad idea. Explain why. (Hint: Can lead to some errors not being caught.)