CSE 3341, Assignment #3
Due: Feb. 28, '20.
20 points.

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). In the object-oriented approach to the CORE interpreter, when dealing with the `Id` class, we said that the array of pointers/references to existing identifiers (or whatever other structure, in place of an array, that we might use to maintain these pointers/references) must be `static`.
   This is a two-part question. First, explain why this array/other structure must be `static`;
   Second, where will space for this structure would be allocated when the CORE interpreter is running? Would it be on the heap or the stack or elsewhere? Explain precisely.