Dylan Parsons
CSCI 2320
11/26/17
Assignment 4

1.
Inheritance and polymorphism are concepts that allow flexibility of methods and variables among various functions. Inheritance is the ability of one class to make use of methods and variables from another class without requiring duplicate declaration or definition. For instance, a class Language may have some variable Grammar, and some function Parse. If the class English inherits from Language, it can then call Parse and use Grammar for itself. Polymorphism follows necessarily from inheritance; the inheriting class, in this case English, may have a method called Parse, with its own definition, parameters, etc. Polymorphism is the option to override a superclass' definition of a function in a subclass.

2.
In imperative programming, polymorphism appears as function overloading, which is the ability to have two or more functions of the same name, as long as the function type and/or parameters change. So, you could have:

```
int printInput(int a)
int printInput(double b)
void printInput(int a)
```

as valid individual functions. Polymorphism in OOP is as described above. Of course, there is no sense of classes in languages like C, so the portion of polymorphism that stems from inheritance is not present in imperative languages.

3.
Abstraction is the idea of removing complicated implementation and non-vital details from the user of a class or method. Only essential information is left accessible/visible, and the rest is abstracted away. For example, language libraries abstract data structures (e.g., sets, stacks, sorting methods) by providing high-level access that removes complicated interactions Encapsulation is effectively a way of achieving a certain level of abstraction. Encapsulation is the hiding of the representation of data, often by providing methods rather than direct access to class data (e.g., get and set)