// Data Structures Laboratory

// Section 004

// Lab 4

// 9/30/2019

// Group 6

// Group Members: Anh Nguyen and Michael Stephens

In this class, we learned about inheritance and polymorphism, 2 very important concept in Object Oriented Programming. If we create a class name Animal to describe an object, like Ants, its member functions and variables will be used to describe Ants. Yet however, Ants are also Insects, and in Insects there are other attributes that could describe Ants, yet wouldn’t be universal in all animals. Therefore, to create a usable class, we can create a derived class Insects, which inherits all of members from Animals, and also include unique attributes to describe Ants.

Since we’re starting to design multiple levels of inheritance / many classes derived from each others, it’s useful to have some of the member functions morphs according to the need of the user to describe the object. Therefore in C++, it’s great that you can override member functions, polymorphism is an important concept in programming.

2.

T1: For base class:

* Since we declared all the derived class access as “Public”, all public members are accessible: getNumplayers, setNumplayers, getScore, setScore
* Play() is declared virtual, thus it will get overridden
* Winner() is not declared virtual, thus it will bind statically, will not be overridden.
* All private members aren’t available to derived class (Score and Numplayers)

T2: For derived class:

* Score and Numplayers won’t be available to these objects
* Difficulty and Boardsize will be available to respective objects
* play() will override the base class’ play()
* winner() won’t override the base class’ winner()

T3: The behaviors of these members are as expected. If we just declare instances of the derived class, we get access to the public members of the base class. We do note that if we pass instance of derived class declared as base class through an outside function, play() override the base class’ function and winner() doesn’t