Individual Summary

Much of what was done in the project required the use of classes. All of the functions used to make calculations and the variables that are used in these functions or represent traits of species belong to classes. There is an Ecosystem class that can store Species objects. This made sense for analyzing populations because the class relationships model how ecosystems have a unique set of species, each of which has its own characteristics. A new concept related to class used in this program was inheritance. Two classes in the program, IndependentSpecies and PredatorPrey, are derived classes of Species and are used to calculate population sizes at different times. Without these classes, all member functions and data fields would have to be included in the Species class, which would make that class very complicated. However, it wouldn’t make sense to make the IndependentSpecies and PredatorPrey classes completely unrelated to Species since the data fields and member functions in the Species class are useful, and sometimes necessary (for example, the data field size) for these classes. Polymorphism is used in the allSpecies vector in the Ecosystem class, which now holds pointers, not objects, of type Species and also stores IndependentSpecies and PredatorPrey pointers since they are derived classes. This required the addition of copy constructors for use in the functions that add pointers to the vector. A function in the Species class was made virtual to support polymorphism. The benefit of this is that my program can now read multiple Species objects from the same file. It couldn’t do this when the table-printing functions took parameters of the IndependentSpecies or PredatorPrey instead of type Ecosystem. In addition to classes, the program uses file input, including using ios::in to open a file for input. The user types in the name of the file that has the information for data field values. It also can create text files or add new data to them using ios::out|app. The project also uses vectors, both to store species in the Ecosystem class and to store population sizes for calculating minimum, maximum, and average population sizes. This is in addition to the original requirements so that the user does not have to search for or calculate these values, which could be difficult for large values of time.

A few other changes were made from the original design. In the Species class, the qAlleleFrequency data field is not used and its value is instead derived from pAlleleFrequency. This simplifies the class and prevents the user from entering invalid values for pAlleleFrequency and qAlleleFrequency, which should add up to one. A setTime() and a setSize() function were added in order for the for loops used in displaying the tables for population sizes to work. A setPAlleleFrequency() function was also added so a user can change that value if needed. A few name changes were also made to reduce confusion. For example, getPopulationSize() was changed to calculatePopulationSize() since it actually recalculates and changes the size. The PredatorPrey class was probably the hardest part of the project to design and implement. The limitation that (predatorDeathRate / gamma) should be close to size and (preyGrowthRate / alpha) should be close to PredatorSize proved to be a challenge. At first, some clearly incorrect values for population sizes were given. Because of this, the private member function isValid() was added to check if this assumption is accurate before displaying the population at the specified time. If it is not the program displays “invalid values”. Two other private member functions, getK() and getPhi() were added to make the calculations for constants used in the equations for finding the predator and prey populations.

This project made several assumptions so improving the program would probably involve adjusting for these assumptions. For calculating population sizes, an improved version would use several different models, instead of just two, and the program would decide which one to use based on the input given. This could help with the problem in the PredatorPrey class of only working for certain cases and also allow the program to take in account other factors that affect population sizes such as competition between species or human influence. Similarly, the equations in the Species class would adjust for populations experiencing change in genetic makeup and are not in equilibrium.