Our project was designed using an Object-Oriented approach which allowed us to implement a pipes and filters technique the best. In our project we decided to make a game chest program using C++ which allowed the user to play a number of games from within the same program. Using pipes and filters allowed us to setup our programs to run off of a small set of inputs from the user. One of the games in our game chest available to be played is Greedy Snake, which runs off a small set of inputs available to the user. It uses these inputs to make a change on the game board that will then be shown to the user with the updated input. By using pipes and filters it allowed us to filter out which game the player wanted to play very easily. An object-oriented approach really only allowed us to go with pipes and filters as a software architecture because object-oriented uses classes and functions that take inputs and return outputs based on the inputs. With that being said it wasn’t a disadvantage to us when it came to designing the software because it was more of an advantage to us in regard of how it allowed us to break down what we needed the program to do and design function and classes around them. Once we had the proper functions and classes setup we could then figure out how we wanted “pipes” to be connect to each function and class. Combining both approaches allowed us to setup functions that would act as filters to make sure bad input never happened and to make sure the good input was passed to the correct functions. Overall the decision to make our project Object-Oriented and setting it up with a pipes and filters architecture was the best thing we could have to done to make sure the project progress easily and each part didn’t get to messy with different component working with each other.