Juhwan Lee
Professor Karla Fant
CS202
November 16th 2020

Program #3 Efficiency Write-up

Program 3 was to create a program that uses operator overloading. Operator overloading allows us to use the operator as a built-in data type for our class object. The purpose was to create a monster battle using Operator Overloading. There are four monster options, and users can create the monster they want, and choose a monster to compete with their opponents. The types of Halloween monsters I have chosen are vampire, witch, werewolf, and devil. The user can create one of the four and as it is created it is stored in the binary search tree. The binary search tree provides five basic functions; Insert, display, retrieve, remove, remove all. And after creating a monster, the user can choose one of the monsters stored in the binary search tree and battle the opponent with the monster. After completing the monster selection, the user proceeds to the section of selecting the attack option and the defense option, each of which has three options. Attack options and defense options are stored in the array of doubly linked lists, which are programmed to start with initial values when starting the program, not user input. It is not used in the client program, but by default, the array of doubly linked lists provides all five basic functions; insert, display, retrieve, remove, and remove all. The way how it is stored is for example, array 1 is a vampire attack option, array 2 is a vampire defense option, array 3 is a witch attack option, and array 4 is a witch defense option and so on. The user selects one of the attack options and one of the defense options, and faces a randomly generated opponent monster. To balance winning and losing, attack option 1 can beat attack option 3, attack option 3 can beat attack option 2, and attack option 2 can beat attack option 1. Defensive options are the same. In this way, the user's attack option and the opponent's attack option are compared with operator overloading, and the user's defense option and the opponent's defense option are also compared with operator overloading, and if both of them win the opponent, the battle is won. Operator overloading was used in binary search tree functions (especially insert function), and in the monster battle part of the client program, and dynamic casting was used a lot in the array of doubly linked lists. For example, it was used in the part to find out the user's monster type in the wrapper function because the array to be accessed is different according to the user's monster type.