S6 Nim Project Write-up

The Nim project has been a very illuminating experience. It has shown us that we can, when pushed, be creative and that, with enough perseverance, can code simple games. We, as a group, found the Nim game to be quite challenging to code. We were each tasked with writing different parts of the code, and we had to break them down and research what needed to be done to complete those tasks. This presented an additional challenge of coordinating each part of the program, method structure, variables, visibility, etc., to ensure each individual effort could easily be incorporated into the final product. Some of the parts of code for the Nim game weren’t as intuitive as other parts. Specifically, the Nim addition and the Computer’s turn in the game required some careful thought in order to get them to function correctly.

The Computer’s turn is very involved. It involves making a list of integers, converting each integer into binary, comparing the binary strings, and then deciding from which list the “sticks” should be removed and removing the correct number of “sticks” so that we get all zeros when the binary strings are added without carrying. At first glance, each of us had to take a step back, exclaim, “What?” and then start thinking. Tracy pushed through to figure out how to make the lists and convert the integers to binary and back again. While he was taking on that challenge, Don started comparing the examples on Nim addition given in the assignment prompt. He began to see some of what needed to be done to replicate it in the program using code. Nim addition is unlike any kind of math problem any of us had seen and it took some effort to find what was required in order to do it so that the addition could be implemented in the code. After we all had a firmer understanding of what Nim addition is and how it is accomplished, Tracy got to work on turning it into code.

For the most part, the challenge was in designing the program, or rather, breaking down the tasks in order to have a clear understanding of what needed to be done in each step of the program. The implementation of the design was much easier once we had a clear picture of how the task needed to be achieved. Some of the other challenges that weren’t as difficult, but still required careful thought were: error checking, building the interface, and arranging the code to be more readable.