Emilio Kiryakos and Lukas Venckus

11/30/23

CS 476

Final Report Summary

We accomplished being able to add list indexing syntax from the Python programming language onto C++ which were the negative indexing such as “arr[-1]”, range indexing for positive and negative range indexing within an array such as “arr[1:3]” and “arr[-3:-5]” respectively and error checking for the indexes that didn’t exist within the array which would then return NULL. The range indexing also can return the reverse of the list when using negative indexes and can return the list forwards with positive indices. We weren’t able to have a clean way of having the user realize out of bounds indices which makes it difficult for the client to catch errors early on. The only way the client would realize that they had gone out of bounds is if they actually got a SEGFAULT during runtime which in most cases isn’t the most desirable outcome. What we wish we could’ve done is have some way of the compiler catching the issue before the program even ran, but we couldn’t figure out how to do that. The most hardest parts of the project were figuring out how to show the range operator in a natural fashion but we just went around this by using a string as the parameter and just parsing the string in an overloaded “[]” operator function. Also returning NULL for just the regular individual integer parameter function was actually kind of difficult because it would not compile regularly by just returning null but I had to set a pointer to null and just return that. Also going from the code to the rules was fairly difficult since we had such specific cases for when things worked and didn’t work, so it was difficult to try to write that out as a mathematical proof. The most interesting parts of the

projects were trying to make the range operator work and deciding on the cases that would work and what cases to just not allow. Also having the range operator actually return another true python_cplusplus_array made it more interesting because the user can get subarrays that have all of the same functionality as the original python_cplusplus_arrays.