DrawLab Project Part 2 Report

This report is about challenges we faced and how we addressed them during the second part of DrawLab project.

Arithmetic Expression Confusion

In the 1st part of the project, in lex file, we write INT as ({MINUS}|{PLUS})?{DIGIT}+, which means we captured integers with a sign before them. This caused, for example, the expression “7-3” was interpreted as stream INT INT by the lexer since both 7 and -3 were interpreted as INT. However, this was not the right interpretation for this expression. Therefore, we changed our lex file so that now it interprets signs and integers separately. For the expression “7-3”, it crates token stream INT MINUS INT. Moreover, to recognize integers with a sing before them, we added the rule integer: INT | MINUS INT | PLUS INT to the parser. The same approach was implemented for floats as well.

Use before Definition Error

If a variable, function or newObject definition is used before its definition, then we needed to give an error. To make parser handle this error, lex file should give the names of “ID”s it encounters instead of only token “ID” to the parser. So we changed the lex file we provided in the 1st part of the project so that it gives string that is recognized as an ID token in “yylval” variable to the parser alongside the token itself. In addition, we needed to make the parser recognize predefined function and object names without explicitly defining them in user code. To achieve this, we added all the predefined function and objects names to appropriate symbol table in the main function before starting parsing.

Multiple Error Catching

We used “error” keyword that is provided by yacc to achieve multiple error catching at one time. We created some rules for some mistakes which can be made by the user and we passed the error messages with more elaborate explanation to “yyerror” function for each of these possible errors. Other than these possible ones, rest of the errors are called “syntax error”. For all the errors in the program, parser provides line number to the user along with type of the error.

Information about Parsing

When parsing is finished, we wanted to provide general information about success or the failures of the process to the user. We added an integer variable called “errors” to yacc file and initialized it to 0 in the main function. Each time “yyerror” is called, “errors” is incremented by 1. So at the end of the parsing process we could show total number of errors to the user if there is any. If “errors” is zero at the end of the process, the parser simply outputs “Parsing completed successfully”.