Compiler Construction

BPDC

(Lab - 02)

1 Designing a Lexical Analyser for a mini-compiler

You are given 3 mini-compilers (in 3 different folders) each with a set of 3-4 sample input C files. Compile the input files using each of those given corresponding mini-compilers, note the outputs in your lab journal with proper reasoning. Run the following sequence of commands in order to run the programs.

1. Run *ls* command to begin with. You should see two files project.l (lex file) and project.y (yacc file) along with the input files.
2. Run *flex project.l*. This should generate the C program for the lexical analyser *lex.yy.c* (run *ls* command and verify whether the file being generated or not).
3. Run *yacc −dv project.y*. This should generate the C program for the syntax analyser *y.tab.c* along with supplementary files *y.tab.h* and *y.output*(run *ls* command and verify whether the files being generated or not).
4. Run *gcc −o mini-compiler lex.yy.c y.tab.c −lfl* to generate the executable.
5. Feed in the C file to your mini-compiler by running *./mini-compiler < input.c*.

Read through the programs *project.l* and *project.y* and justify your output with concise proper reasoning.

1.1 Modify your lex program to incorporate the following changes

In this part, you are supposed to modify the third exercise to incorporate the following changes.

1. As per the current set up, the programmer is supposed to use only those lower case alphabets in variable names in their C program. Modify your lex program so as to let the programmer have uppercase letters A to F together with digits 0 to 9 in variable names.
2. Add provision to declare variables of type float, double and char.
3. Terminate your program with an error message if in case the programmer uses keywords "if", "while", "do", and "for" as variable names. Note that its permitted to have variable names beginning with keywords (ifvar, donut etc.) (hint: rely on conflict resolution rules).

1.2 Type checking in program statements

In this problem (files in the folder *problem*), you are supposed to incorporate two functions *insert_to_table*() and *lookup_in_table*() to your yacc program (prototype of the functions and symbol table data structure are given. You just have to write the body of both functions) so that in each of those program statements, the parser would confirm the following:

1. Whether all variables in an expression are of the same type. If not, would show up a type mismatch error message (as in file input3.c the error message should be: ‘type mismatch in the expression’).
2. Is there a variable declared multiple times (show a multiple declaration error message and terminate) (input2.c, the error message should be: ‘multiple declaration of variable vartwo’). This has to be taken care of, in the function *insert_to_table*() checking whether the variable is already there in the table, before making an entry.
3. Is there an undeclared variable (show an undeclared error message and exit) (in input4.c the error message should be: ‘variable “varfour” undeclared’). The wrapper program partially takes care of it. You just have to implement the *lookup_in_table*() function appropriately.