CSL 355. Programming Assignment 2
Due date: next Thursday, 30th March by midnight

Instructions.

1. Do in groups of 4-5.
2. 30% penalty for late submission.
3. You have to submit a single C program.

1 Basic C

As discussed in class, *basic C* language supports finite precision constants, variables, semi-infinite arrays, expressions, assignments, and while loops:

1. A *program* consists of a sequence of statements separated by semi-colons.
2. A *statement* is either an *assignment* or a *while* loop.
3. An assignment is a statement of the form $V = E$; where V is a variable and E is an expression.
4. A while loop is a statement of the form $while(E)\{S\}$; where E is an expression and S is a sequence of statements separated by semi-colons.
5. Expressions are obtained by applying binary operators $*, /, +, -, <, ==$ over variables and constants. Among the binary operators, $<, ==$ have lower precedence than $+, -$ and $+, -$ have lower precedence than $*, /$. Expressions can also be parenthesized using $(,)$. All binary operators associate from left-to-right.
6. Variable names are from $\{a-z\}^+$ except for "while" which is a keyword. Given variable v and a non-negative integer i, $v[i]$ denotes the i-th entry of integer array $v[]$. v itself denotes $v[0]$. Variables are declared the first time they are used and take only finite precision constants as values.
7. In addition to variables, the language also has support for *finite precision constants* with at most B digits after the decimal point. These constants are of the form $\{0-9\}^+$ or of the form $\{0-9\}^+.\{0-9\}^k$ where $k \leq B$.

2 Questions

Question 1. Give an unambiguous grammar G for basic C with finite precision B. Construct a deterministic tokenizer-parser which constructs the unique syntax tree in grammar G for an input program in basic C language.

Your program should output the syntax tree in generalised list format where each list consists of a vertex followed by the lists of its children ordered from

left to right. Use $\backslash, /$ as left and right parentheses for the output generalised list. (Note that $(,), \{, \}, [,]$ cannot be used as they are already part of basic C syntax.)