Planning:

For the compiler, I would be using gprolog provided by secure shell. Regarding how to solve this, I am going to break down into multiple tasks. One of the tasks would be string manipulation. That is how to handle the input/output. Another task would be to convert the CLITE grammar into equivalent prolog grammar. One of the problems that I might face doing this would be converting EBNF grammar into BNF grammar. I will solve this problem by looking at my theoretical concept notes. Another problem that I might face would be right those rules in proper prolog format. I will be solving this problem by familiarizing with the syntax of the prolog language. I will be learning the syntax of the language through a website called [www.gprolog.com](http://www.gprolog.com).

Design:

With the help of theoretical concept notes, I converted all the EBNF grammar into BNF equivalent prolog rules. For example, I am going to change Declarations :- { Declaration } into the following statements:

declarations(A-A1):- declaration(A-A1).

declarations(A-A1):- declaration(A-A2), declarations(A2-A1).

Conversion of EBNF to BNF is mostly done using the concept of tail recursion in the case of kleene star operation. For other operation, simple concept should be sufficient. For example,

The following statement

Assignment :- Identifier [ [ Expression ] ] = Expression ; can be changed into the following:

assignment:- identifier, aequals, expression, semicolon.

assignment(A-A1):- identifier(A-A2), square(A2-A3), expression(A3-A4), clsquare(A4-A5), aequals(A5-A6), expression(A6-A7), semicolon(A7-A1).

aequals(['='|A]-A).

Implementation:

Testing:

Once I was done with the project, to make sure whether I had satisfied all the requirement, I tested my program with bunch of test cases.

input1(A):- program([int, main, '(', ')', '{',int,a,b,c,d,e,f,;,int, g,;,a,=,!,b,+,c,==,d,>,e,&&,f,;,'}']-[]).

%input2(A):- program(['int main () {',int,a,;,b,=,1,;,x,=,x,+,1,;,'}']-[]).

%input4(A):- program(['int main () {',int, a,;,a,=,a,+,1,;,'}']-[]).

%input5(A):- program(['int main () {',int,a,;,x,=,x,+,1,;,'}']-[]).

The following above were my few test cases. Here all of the test cases were the CLITE program.

Also, I test with some not CLITE program. And all of the test cases passed the test.

That means all the CLITE program that were well defined printed out saying that they were well defined. And the inputs which were not CLITE program just printed no as their output, which meant that they were not the CLITE program.

Post Mortem:

One of the problem with this lab is that I couldn’t a good way to represent all the letters and characters into few lines. I had to write the entire letters which means I had to write separately from a to z.

Another problem with this lab is that I had to separate the input string into a list which is separated by commas.

One good aspect about this lab is that now I have a good understanding of prolog language and its logic. Also, I thoroughly learned how to convert EBNF into BNF grammar. Another good aspect about this lab is that I learnt about the concept of parsing and scanning clearly. Previously I didn’t know what parsing was.

Another good aspect about this lab is that I learnt a new programming paradigm, i.e. functional programming.