**AIM:**

To design an online simulator that implements a Parser (Non Recursive Descent- parser) for the given grammar.

**PROGRAMMING LANGUAGES USED:**

* C Programming Language
* HTML
* CSS
* JavaScript

**THEORY:**

A parser generator is a good tool that you should make part of your toolbox. A parser generator takes a grammar as input and automatically generates source code that can parse streams of characters using the grammar..

The generated code is a parser, which takes a sequence of characters and tries to match the sequence against the grammar. The parser typically produces a parse tree, which shows how grammar productions are expanded into a sentence that matches the character sequence. The root of the parse tree is the starting nonterminal of the grammar. Each node of the parse tree expands into one production of the grammar. We’ll see how a parse tree actually looks in the next section.

The final step of parsing is to do something useful with this parse tree. We are going to translate it into a value of a recursive data type. Recursive abstract data types are often used to represent an expression in a language, like HTML, or Markdown, or Java, or algebraic expressions. A recursive abstract data type that represents a language expression is called an abstract syntax tree (AST).

**SIMULATOR LOGIC:**

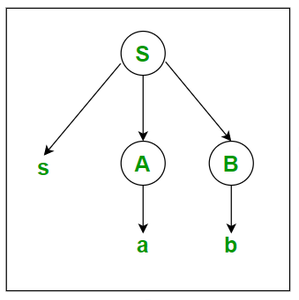
Let us take an example of a Grammar (Production Rules).

S -> sAB

A -> a

B -> b

The input string is “sab”, then the Parse Tree is :

[](https://media.geeksforgeeks.org/wp-content/uploads/20200913131917/cd2-299x300.PNG)

**HANDLING ERRORS:**

By default, Antlr parsers print errors to the console. In order to make the parser modular, however, we need to handle those errors differently. You can attach an ErrorListener to the lexer and parser in order to throw an exception when an error is encountered during parsing. The Sum.g4 file defines a method reportErrorsAsExceptions() which does this. So if you copy the technique used in this grammar file, you can call:

lexer.reportErrorsAsExceptions();

parser.reportErrorsAsExceptions();

right after you create the lexer and parser. Then when you call parser.root(), it will throw an exception as soon as it encounters something that it can’t match.

**DISCUSSION:**

We wrote the online simulator using C programming language and integrated it with the website designed using HTML, CSS, JavaScript. The simulator takes a Grammar(Production Rules) as user input and generate the corresponding parse trees according to the grammar/ production rules.