**CS323 Assignment Documentation**

**1.**    **Problem Statement**

What is expected out of this assignment is to build a parser, and use the lexer from the first part. Then to use the parser to analyze the tokens and output the according production rules.

**2.**    **How to use your program**

Download the zip, the unzip it. Locate where the debug folder is then using the command prompt screen use cd to reach the directory where the debug file is located. Then go into the debug folder using the command prompt, after that type Parser.exe<test1.txt to run the program.

**3.**    **Design of your program**

We used 2D array for the grammar matrix and its dimensions are 15 rows by 19 columns. We also used a struct for the Symbol instead of a class, along with making the grammar matrix a struct too. There are also a few functions inside the Parser class that are used:

* getTokenId – returns the token id of the given corresponding token string
* getAbbreaviaiton – returns the abbreviation of the nonterminal
* isTerminal – returns true is the symbol is terminal and false otherwise
* equals – returns true if the token matches the symbol
* START\_SYMBOL – a static start symbol with the name “Pgm”
* END\_MARKER – a static end marker with the name “$”
* getRow – returns the row index based on the symbol
* getColumn – returns the column index based on the symbol
* get – gets the production based on the symbol row and token col
* getColumnHeader – get the header of a column based on a token
* push\_symbol – pushes a symbol from the stack of symbols
* pop\_symbol – pops a symbol from the stack of symbols
* print\_trace – prints traces of the syntactic symbols
* print\_error – prints an error message
* Parser – class constructor
* Start\_parse – starts the syntactic analysis

**4.**    **Any Limitation**

None.

**5.**    **Any shortcomings**

Does not parse correctly a fatal error occurs.

Test Cases

**Test case 1**

integer a#, b, c#d, efgh#ij;

integer low = 22;

integer high = 40;

float number;

read(float);

----------------------------------------------------------------------------------------------------

**Test case 2**

%%

if ( low > float )

{

write(float);

}

fi

----------------------------------------------------------------------------------------------------

**Test Case 3**

if (low < high)

{

a# = low + high;

c#d = high - low;

}

fi

if ( low > high)

{

c#d = low - high;

efgh#ij = low \* high;

}

fi

return 0;