1 Introduction

Previously, you had designed the grammar for your parser. Now you will be implementing your parser in code following your own grammar constraints. The Parser shall take the token-lexeme pairs generated by the Lex, and analyze its syntax according to the grammar. If it encounters incorrect syntax or erroneous statements, it shall gracefully halt execution and print an appropriate error message. In the case of completely correct syntax, it shall produce a parse tree of the code. It shall also add to the symbol table the datatypes of all identifiers.

The language is once again as follows:

```
int: num;
char: my_char;

// lets assign variable my_char a value
my_char = 'd';
print("my char contains: ");
println(my_char);

/*
    The program here onwards is an iterative algorithm
    for fibonacci numbers
*/

println("enter a number");
input -> num;
int: a=0, b=1, c=0;
println("The fibonacci seq is: ");
println(a);
println(b);
while c < num:
{
    int: temp = a+b;
    a = b;
    b = temp;
    println(temp);
    c++;
}
```

You can copy this code in your own file or use `sample_code.cc` file for the assignment.

The .cc extension is arbitrary, it means compiler construction and it shall be our designated extension for our language. Here are our language's features that you will have to implement:

1. Currently only 2 data types are supported:
   (a) int
   (b) char
2. Several keywords are absolutely necessary:
   (a) Decision Statements: if, elif, else
   (b) Looping Statements: while
   (c) Standard input statement: input
   (d) Standard output statement: print and println
3. How can we forget arithmetic: + - * /
4. Relational operators are just as important: < <= > >= == ~=
5. Comment your heart out:
   (a) Single line comments using double back slash \\
   (b) Multi line comments using /* */
6. Identifiers start with a character followed by any number of characters, digits or the underscore symbol
7. The language also supports:
   (a) Numeric constants: only integers
   (b) Literal constants: only a single character enclosed in single quotes eg 'a'
   (c) Strings: sequence of characters and white spaces enclosed in double quotes
8. Operators make our lives easier:
   (a) Parenthesis, braces and square brackets
   (b) Assignment operator =
   (c) Input operator ->
   (d) colon, semi colons and commas are also required

2 Task

Code your parser pertaining to the grammar defined previously. Run the lexer and parser in continuation on the sample code generating a parse tree and a symbol table that are written to text files. Do not skip running the lexer, it is important that you link the lexer to the parser using an appropriate mechanism.

Here's an unrelated sample parse tree, it is fashioned after the example in the slides. Here all Xs will be replaced by appropriate tokens. Refer to the slides for more details.

```
X
|__X
|__X
: |__X
: |__X
: : |__^
|__ID (sum)
|__X
: |__X
: |__X
: : |__ID (i)
: : |__A'
: : : |__COMMA
: : : |__D
: : : : |__INT
...
: : : : : |__X
: : : : : : |__^
: : : |__X
: : : : |__^
: : |__SEMICOLON
: |__X
|__X
: |__^
```

3 Submission Instructions

The submission guideline is very straight forward. Non adherence might result in marks deduction, hence read this section carefully.

- Submissions shall be uploaded compressed in **zip** format.

- All documents shall be submitted in **pdf** format.
- Submissions shall not include executables
- All submissions containing code shall include a README file, it shall provide thorough instructions for compiling and running the code submitted.
- All submitted code shall be able to compile and execute, marks might be deducted otherwise.
- The submitted code shall compile on **Linux** and/or **Windows**. In case the student owns a **Mac**, they shall make sure the libraries/packages they use are posix compliant, which shall make their code compile able on linux.

Lastly, understand that your submissions will thoroughly be checked for plagiarism. Plagiarism will not be tolerated and cased will be met with strict action. Once a case has been established, plea appeals and apologies will not be entertained.

In order to aid you, the TA will be available to you on all days, including the weekend. He may be contacted on his email: `l176304@lhr.nu.edu.pk` or through other channels that he shall share with you soon. Make the most out of this, and approach him instead of resorting to cheating and academic fraud.