Compiler Construction

Special Python Language(SPL)

Assignment 4b **Fall 2022** **Net Weightage: TBD**

Dead Line: 28th November 2022 11:59 PM (Non-extendable)

- **Late submissions will not be accepted for any reason**
- There is no marks for empty submissions
- **You have to implement this in a programming language chosen in Assignment 1**
- Please submit a .zip file, containing the code, report and all the output files. Name the zip file as RollNumber_4b.zip
- **Report should contain all the steps you perform to attain the normalized grammar, as well all the valid assumptions made during the assignments.**
- **Do Not compare this grammar with the actual python syntax as this is our own version, which can vary. You will be evaluated and given the credit for writing a parser for this SPL only.**
- Avoid all sorts of plagiarism as it can have serious consequences
- In case of any queries, feel free to email me at bscs17046@itu.edu.pk
- **ATTENTION: No EXTENSIONS POSSIBLE. Don't wait for the final date start working today.**

Instructions

Consider the following SPL Grammar:

PROG → PROG DEC | DEC

DEC → KEY IDFR (VARDEC) COL BLOCK

COL → :

KEY → def

VARDEC → ε | VARDECNE

VARDECNE → IDFR | VARDECNE, IDFR

BLOCK → { ENE }

ENE → EXP | ε

EXP → IDFR | INTLIT | IDFR = EXP | (EXP BINOP EXP) | IDFR (ARGS) | if EXP then COL BLOCK else COL BLOCK | while EXP COL BLOCK | for IDFR OPLIT (ARGS) COL BLOCK | IDFR = input (QUOTES STRING QUOTES) | print (QUOTES STRING QUOTES) | KEYWORD (INTLIT)

KEYWORD → int | float | str

QUOTES → “

STRING → INTLIT | STRING IDFR

ARGS → ARGSNE | ε

ARGSNE → EXP | ARGSNE, EXP

OPLIT → in | not in

IDFR → id (refer to your lex definition of identifier)

INTLIT → num (refer to your lex definition of numbers)

BINOP → == | < | > | <= | >= | + | * | - | % | / | and | or | !=

Consider this Special Python Language (SPL)

Task 1

Perform manual preprocessing on this grammar to remove all the ambiguities (direct or indirect left recursion of left factoring) to make it an LL(1). Submit a report of your ambiguity removal activity and the normalized grammar. If you need to define more grammar variables (non-terminals) then use meaningful names for them (for instance if y need a new variable in the Block production, then you may use BLOCK1 or BLOCKA). Please don't use X Y Z as your new non-terminals. Submit your grammar in the SPL-Grammar.txt. Follow the grammar format from the assignment 4a.

Task 2

Run your program of 4a to produce the LL (1) parsing table for the SPL-Grammar and store the output in the SPL-Parsing.csv or .xls.

Task 3

Write a driver program for your SPL top down parser. Your program must use the LL (1) parsing table constructed in the task 2 and parse any program written in the SPL language present in the Input.txt file. You will be given a couple of the sample inputs and their respective output for testing your driver program.

Error Handling: You are required to use the panic mode error recovery technique for your program. To implement the error handling you will be required to define the synchronizing tokens for each of the non-terminals. On encountering an error your program must skip the tokens one by one until you get a synchronizing token for your current grammar variable.

Once a synchronizing token is found your program would resume the parsing or halt if the sentinel terminal is reached.