**CPSC 323 - PROJECT ASSIGNMENT 2**

Programming Assignment 2

Project 2 consists of one program to be submitted/uploaded online on Canvas.

You are allowed to write your project in **C/C++/Java/Python** etc. but you ARE NOT allowed to use **Yacc, Bison, or any other items similar** that assists in the creation of compilers.

Given the following CFG and the parsing table, write a program to trace input strings over the alphabet { i, +, - , \*, / ), ( } and **ending with $.**

1. Given the CFG and the Predictive Parsing table below:
   * [60 points] Write a program to trace an input string given by the user. Save it as **Prog1** and upload it in canvas (either the zip file or GitHub link). Test your program with the following 3 input strings:

# (1) (a +a)\*a$

**(2) a\*(a/a) $**

# (3) a(a+a) $

* + [20 points] Show the content of the stack implementation / stack flow after each match.
  + [20 points] Report file – Your report file should contain explanation of your code, in-built functions used, important checkpoints [if it is present], explanation should be short and crisp, output screenshot of your code and should not exceed more than 2 pages.

1. Following is the grammar, and parsing table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | FIRST | | FOLLOW |
| E  Q T  R | ( a  + -  ( a  / \* | ɛ ɛ | $ )  $ )  + - ) $  + - ) $ |
| F | ( a |  | + - \* / )  $ |

|  |  |  |
| --- | --- | --- |
| **Given CFG** | **CFG**  **after removing**  **left-recursion rules** | **First and Follow table** |
| **E** → **E+T E** → **E- T**  **E** → **T**  **T** → **T\*F T** →**T/F T** → **F**  **F** →**(E )**  **F** → **a** | **E** → **TQ Q** → **+TQ Q** → **-TQ**  **Q** → ɛ  **T** → **F R**  **R** → **\*FR R** → **/FR R** → ɛ  **F** → **( E )** |  |
|  | **F** → **a** |

# Predictive parsing table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| state  s | a | + | - | \* | / | ( | ) | $ |
| E | TQ |  |  |  |  | TQ |  |  |
| Q |  | +TQ | -TQ |  |  |  | ɛ | ɛ |
| T | FR |  |  |  |  | FR |  |  |
| R |  | ɛ | ɛ | \*FR | /FR |  | ɛ | ɛ |
| F | a |  |  |  |  | (E) |  |  |

1. **Output :**

For the same grammar and parsing table if the input string is (a+a) $, then **Output** must be displayed like this along the stack implementation ( whole stack flow should be shown, though in the example only the end of the stack is shown) Example,

# Input: (a+a) $

**Stack: [‘$’, ‘Q’, ‘ R’]**

# Output: String is accepted/ valid.

**Input: (a+a) e $ Stack : [‘$’, ‘Q’ ,R’]**

# Output: String is not accepted/ In valid.