Isaac Perez

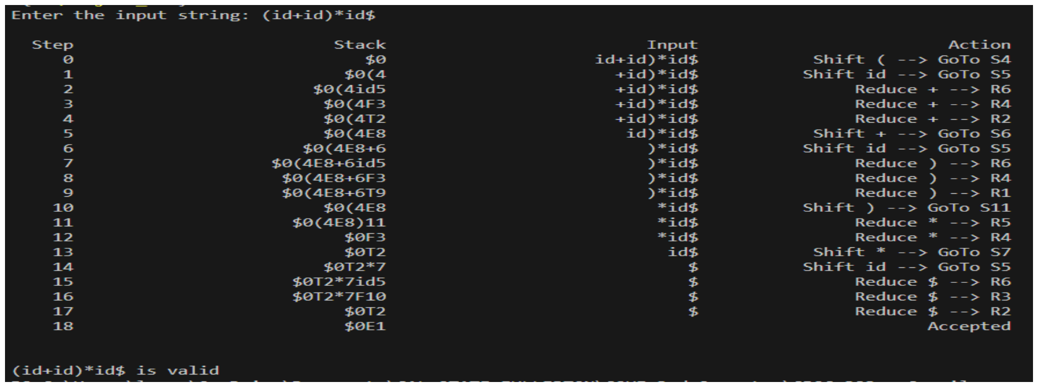
Darren Cruz

Hao Ly

CPSC 323

CPSC 323 Project 02 Documentation

- **Program Name:** Parser
- **Goal:** Our program implement a parser for context-free grammar (CFG). The parser trace input strings over a set of symbols { id, +, *,), (}, ending with a special marker "$". The primary goals and functionalities include parsing strings, stack implementation, and letting the user know is the input valid or not
- **Programming Language:** C++
- **Roles:**
    - Isaac Perez: Design the parse tree for each case
    - Darren Cruz: Implement the program
    - Hao Ly: Implement the program & documentation of the code
- **Input String:** (1) (id+id)*id$ (2) id*id$ (3) (id*)$
- **Output:** Stack Implementation & String is accepted / String is not accepted
    - (1) (id+id)*id$

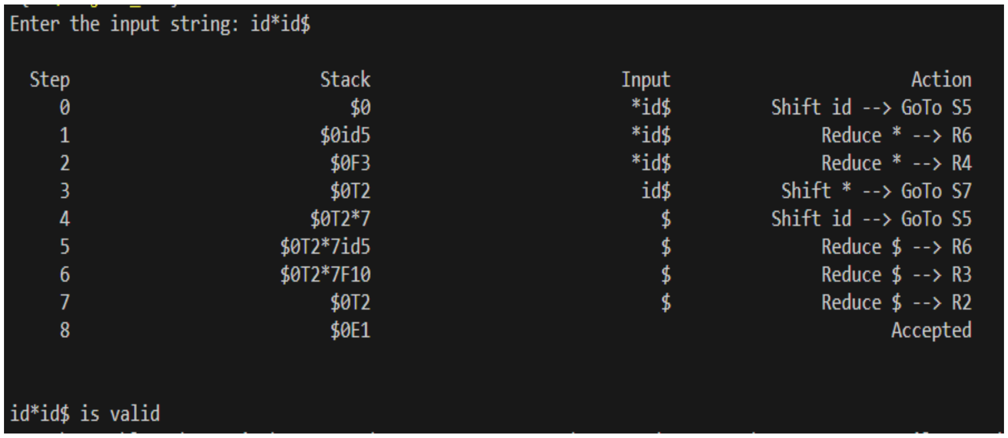


```
Enter the input string: (id+id)*id$

  Step                         Stack                        Input                     Action
     0                            $0                   id+id)*id$          Shift ( --> GoTo S4
     1                          $0(4                    +id)*id$         Shift id --> GoTo S5
     2                       $0(4id5                    +id)*id$             Reduce + --> R6
     3                        $0(4F3                    +id)*id$             Reduce + --> R4
     4                        $0(4T2                    +id)*id$             Reduce + --> R2
     5                        $0(4E8                     id)*id$          Shift + --> GoTo S6
     6                      $0(4E8+6                       )*id$         Shift id --> GoTo S5
     7                   $0(4E8+6id5                       )*id$             Reduce ) --> R6
     8                    $0(4E8+6F3                       )*id$             Reduce ) --> R4
     9                    $0(4E8+6T9                       )*id$             Reduce ) --> R1
    10                        $0(4E8                        *id$        Shift ) --> GoTo S11
    11                     $0(4E8)11                        *id$             Reduce * --> R5
    12                          $0F3                        *id$             Reduce * --> R4
    13                          $0T2                         id$          Shift * --> GoTo S7
    14                        $0T2*7                           $         Shift id --> GoTo S5
    15                     $0T2*7id5                           $             Reduce $ --> R6
    16                     $0T2*7F10                           $             Reduce $ --> R3
    17                          $0T2                           $             Reduce $ --> R2
    18                          $0E1                                                 Accepted


(id+id)*id$ is valid
```

- (2) id*id$

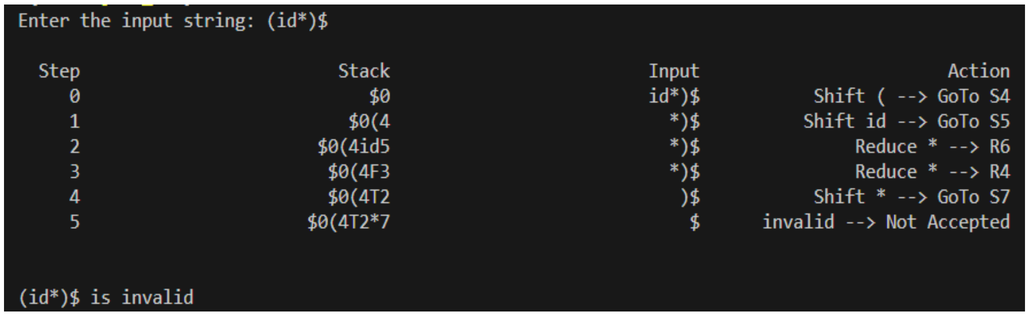


```
Enter the input string: id*id$

  Step                      Stack                     Input            Action
     0                         $0                      *id$     Shift id --> GoTo S5
     1                      $0id5                      *id$        Reduce * --> R6
     2                       $0F3                      *id$        Reduce * --> R4
     3                       $0T2                       id$     Shift * --> GoTo S7
     4                     $0T2*7                         $     Shift id --> GoTo S5
     5                  $0T2*7id5                         $        Reduce $ --> R6
     6                  $0T2*7F10                         $        Reduce $ --> R3
     7                       $0T2                         $        Reduce $ --> R2
     8                       $0E1                                      Accepted


id*id$ is valid
```

- (3) (id*)$



```
Enter the input string: (id*)$

  Step                      Stack                     Input            Action
     0                         $0                   id*)$      Shift ( --> GoTo S4
     1                       $0(4                     *)$     Shift id --> GoTo S5
     2                    $0(4id5                     *)$         Reduce * --> R6
     3                     $0(4F3                     *)$         Reduce * --> R4
     4                     $0(4T2                      )$      Shift * --> GoTo S7
     5                   $0(4T2*7                       $  invalid --> Not Accepted


(id*)$ is invalid
```

- **Analyze:**
  - **Parsing Table and Entry Struct:**
    - The parsingTable is a vector of vectors of Entry structures. Each entry contains an input symbol and the corresponding action (shift, reduce, accept).
    - The Entry structure has two string members: input and action.
  - **Parser Class:**

    - The Parser class contains a private member parsingTable and several functions to parse the input string.
    - The stackToString function converts the content of the stack to a string for display purposes.

- **Parsing Logic:**
    - The parseInput function performs the parsing of the input string using a stack and the parsing table.
    - The function iterates through the input string, updating the stack and displaying the step-by-step process.
    - It handles actions such as shifting, reducing, and accepting based on the parsing table entries.

- **Main Function:**
    - The main function creates an instance of the Parser class and takes user input for the input string.
    - It then calls the parseInput function to initiate the parsing process.

- **Output:**
    - The output is formatted to display the step number, stack content, input remaining, and the action taken at each step.
    - The program outputs messages indicating whether the input string is valid or invalid based on the parsing process.

- **Setup & Run:**
  1) **Clone from github:**

     Our program is hosted in a repository on GitHub and here is its link

     https://github.com/lyquochao84/CPSC323_Project02

  2) **Create a folder:**

     The user has to create a folder in their local computer before cloning the project from GitHub to store all the program files

  3) **Install Visual Studio Code:**

     Install Visual Studio Code before you use the program

  4) **Open in Visual Studio Code:**

     After cloning the project and downloading the VSCode, the user now can open it on Visual Studio Code

  5) **Install all C++ Extension In VSCode:**

     Visual Studio Code will ask the user to install everything the user needs to run the C++ program. Follow the VSCode instructions to install C++ extension

  6) **Click "Run Code":**

     After installing the C++ extension, the user can right-click and choose "Run Code" or the user can enter the shortcut command Ctrl + Alt + N

7) **Enter input:**

The program will ask the user to enter any input they want to test

8) **Show Output:**

After the user enters, the program will display the stack implementation for the input string with step, action, input, and action along with whether the output string is valid or not