BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

(BUBT)

Lab Report

Course Code : CSE 324

Course Title : Compiler Design Lab

Date of Submission :

Submitted By

Name : Aktaruzzaman

ID : 21222203031

Intake : 41

Section : 1

Submitted To

Ms. Adeeba Anis

Lecturer

Department of Computer Science & Engineering

Bangladesh University of Business and Technology (BUBT)

Experiment No: 9
Experiment Name: Recursive Descent Parsing Implementation

Problem Structure:
The objective of this experiment is to implement a recursive descent parser for a specific grammar. Recursive descent parsing is a top-down parsing technique where each non-terminal in the grammar is associated with a parsing function. The parser recursively calls these functions to parse the input string according to the grammar rules. The grammar is defined as follows:

E -> TE'
E' -> +TE' | ε
T -> FT'
T' -> *FT' | ε
F -> (E) | a

Procedure
1. Define parsing functions for each non-terminal symbol in the grammar.
2. Implement parsing logic within each function based on the grammar rules.
3. Use a `match` function to compare the current token with the expected token and advance to the next token if they match.
4. Start parsing from the start symbol of the grammar using the `parse` function.
5. If the parsing is successful and the entire input is consumed, output "Accepted"; otherwise, output "Not Accepted".

Code:

recursiveParsing.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;
class Parser {
private:
  string input_string;
    char current_token;
    int index;

    char getNextToken() {
        if (index < input_string.length()) {
            char token = input_string[index];
            index++;
            return token;
        } else {
            return '$'; // $ indicates end of input
        }
    }

    void match(char expected_token) {
        if (current_token == expected_token) {
```

```
            current_token = getNextToken();
        } else {
            cout << "Syntax Error\n";
            exit(1);
        }
    }

    bool E() {
        if (T()) {
            if (E_prime()) {
                return true;
            }
        }
        return false;
    }

    bool E_prime() {
        if (current_token == '+') {
            match('+');
            if (T()) {
                if (E_prime()) {
                    return true;
                }
            }
            return false;
        } else {
            return true;
        }
    }

    bool T() {
        if (F()) {
            if (T_prime()) {
                return true;
            }
        }
        return false;
    }

    bool T_prime() {
        if (current_token == '*') {
            match('*');
            if (F()) {
                if (T_prime()) {
                    return true;
                }
            }
            return false;
        } else {
            return true;
        }
    }

    bool F() {
        if (current_token == '(') {
            match('(');
            if (E()) {
                match(')');
                return true;
            } else {
                return false;
            }
        } else if (current_token == 'a') {
            match('a');
            return true;
        } else {
            return false;
        }
    }

```

```
92  public:
93      Parser(string input_str) : input_string(input_str), index(0) {}
94
95      void parse() {
96          current_token = getNextToken();
97          if (E()) {
98              if (current_token == '$') {
99                  cout << "Accepted\n";
100             } else {
101                 cout << "Not Accepted\n";
102             }
103         } else {
104            cout << "Not Accepted\n";
105         }
106     }
107 };
108
109 int main() {
110     cout << "Please enter your Input: ";
111     string input_string;
112     getline(cin, input_string);
113
114     Parser parser(input_string);
115     parser.parse();
116     return 0;
117 }
```

Input and Output

```
coderaktar@root: ~/Desktop
coderaktar@root:~/Desktop$ chmod +777 recursive_parsing
coderaktar@root:~/Desktop$ ./recursive_parsing
Please enter your Input: a+a*a
Accepted
coderaktar@root:~/Desktop$ ./recursive_parsing
Please enter your Input: a+a/a
Not Accepted
coderaktar@root:~/Desktop$ ./recursive_parsing
Please enter your Input: a*a*a
Accepted
coderaktar@root:~/Desktop$
```

Conclusion: This program implements a recursive descent parser for the given grammar, allowing it to parse and determine whether a given input string is accepted by the grammar.