S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 2

Aim: Implementation of Recursive Descent Parser in C

(a) C Program to construct the recursive descent parser for the following grammar

S->aABb
A->c| ε
B->d| ε

(a) C Program to construct the recursive descent parser for the following grammar

S -> iEtS'|a
S'-> eS|ε
E->b

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: 6th Semester/3rd Year

Academic Session: 2024-25

Date of Performance:

Date of Submission:

Aim: Implementation of Recursive Descent Parser in C :-

(a) C Program to construct the recursive descent parser for the following grammar

S->aABb

A->c| ε

B->d| ε

(b) C Program to construct the recursive descent parser for the following grammar

S -> iEtS'|a

S'-> eS|ε

E->b

OBJECTIVE / EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To illustrate the use of Top down Parser
- To understand, how to demonstrate the Recursive Descent parser.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement:

- Processor: Dual Core
- RAM: 4 GB
- Hard Disk Drive: > 80 GB

Software Requirement:

- C Compiler

THEORY:

1) Top down parsing

2) Working of Recursive descent parser

3) Example

4) Advantages

5) Disadvantages

AIM: (a) C Program to construct the recursive descent parser for the following grammar

S->aABb

A->c| ε

B->d| ε

CODE:

```
#include <stdio.h>
#include <stdlib.h>
/*
Grammar:
S -> aABb
A -> c | ε
B -> d | ε
*/
char l;
int match(char c) {
    if (l == c) {
        l = getchar();
        return 1;
    } else {
        return 0;
    }
}
void B() {
    if (l == 'd') {
        match('d');
    }
}
void A() {
    if (l == 'c') {
        match('c');
    }
}
```

```
void S() {
    if (match('a')) {
        A();
        B();
        if (!match('b')) {
            printf("Error: Expected 'b'\n");
            exit(1);
        }
    } else {
        printf("Error: Expected 'a'\n");
        exit(1);
    }
}
int main() {
    printf("Enter a string ending with $:\n");
    l = getchar();
    S();
    if (l == '$') {
        printf("\nParsing Successful\n");
    } else {
        printf("\nError: Unexpected input '%c' after parsing\n", l);
        exit(1);
    }
    return 0;
}
```

OUTPUT:

```
csc15@linux-p2-1272il:~/CS22130$ ./a.out
Enter a string ending with $:
acdb$

Parsing Successful
csc15@linux-p2-1272il:~/CS22130$
```

AIM: (b) C Program to construct the recursive descent parser for the following grammar

S -> iEtS'|a

S'-> eS|ε

E->b

CODE:

```
#include <stdio.h>
#include <stdlib.h>
char lookahead;
void S();
void S_prime();
void E();
void getNextChar() {
   lookahead = getchar();
}
void match(char expected) {
   if (lookahead == expected) {
      getNextChar();
   } else {
      printf("Syntax error: Expected '%c', but found '%c'\n", expected, lookahead);
      exit(1);  // Exit on error
   }
}
void E() {
   if (lookahead == 'b') {
      match('b');
   } else {
      printf("Syntax error in E: Expected 'b', but found '%c'\n", lookahead);
      exit(1);
   }
}
void S_prime() {
   if (lookahead == 'e') {
```

```
        match('e');
        S();
    }
}
void S() {
    if (lookahead == 'i') {
        match('i');
        E();
        match('t');
        S();
        S_prime();
    } else if (lookahead == 'a') {
        match('a');
    } else {
        printf("Syntax error in S: Expected 'i' or 'a', but found '%c'\n", lookahead);
        exit(1);
    }
}
int main() {
    printf("Enter the input string (end with '$'):\n");

    getNextChar();
    S();
    if (lookahead == '$') {
        printf("Parsing successful!\n");
    } else {
        printf("Invalid input: Expected end of string '$', but found '%c'\n", lookahead);
        exit(1);
    }
    return 0;
}
```

OUTPUT:

```
csc15@linux-p2-1272il:~/CS22130$ ./a.out
Enter the input string (end with '$'):
ibta$
Parsing successful!
csc15@linux-p2-1272il:~/CS22130$
```

DISCUSSION AND VIVA VOCE:

1. What is Top Down parser?
2. How Recursive Descent parser is implemented?
3. What happens to parser if left recursion is present in the grammar?
4. What is advantage and disadvantage of this parser?
5. Which type of derivation does top down parser use? Why?

REFERENCE:

- Principles of Compiler Design, by Alfred Aho and Jeffrey Ullman
- https://iitd.github.io/col728/lec/parsing.html