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

Practical No. 4

AIM: Implement a Program to check whether the given context grammar is LL (1).

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: 6th Semester/3rd Year

Academic Session: 2024-2025

Date of Performance:

Date of Submission:

AIM: Implement a Program to check whether the given context grammar is LL (1).

OBJECTIVE / EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To demonstrate how to check the grammar is LL (1)).
- To compute FIRST and FOLLOW.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement:

- Processor: Dual Core
- RAM: 1GB
- Hard Disk Drive: > 80 GB

THEORY:

1) Details about LL(1) Grammar

2) Methods to check whether the given grammar is LL(1) or not

3) Left Recursion and LL(1) grammar

4) Application of LL(1) grammar

ALGORITHM / PROCEDURE:

A grammar G is LL(1) if and only if the following conditions hold for two distinctive production rules $A \rightarrow \alpha$ and $A \rightarrow \beta$:

1. Both α and β cannot derive strings starting with same terminals.
2. At most one of α and β can derive to ε.
3. If β can derive to ε, then α cannot derive to any string starting with a terminal in FOLLOW(A).

Before applying above conditions, we must have following: -

1)The grammar is free from left recursion.

2)The grammar should not be ambiguous.

3)The grammar has to be left factored in so that the grammar is deterministic grammar.

CODE:

csc15@linux-p2-1272il: ~/CS22130

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

// Check for FIRST/FIRST conflicts
bool hasFirstFirstConflict(char firstA[][10], int n) {
    for (int i = 0; i < n; i++)
        for (int j = i + 1; j < n; j++)
            if (strcmp(firstA[i], firstA[j]) == 0)
                return true;
    return false;
}

// Check for FIRST/FOLLOW conflicts
bool hasFirstFollowConflict(char firstA[][10], int n, char followA[]) {
    for (int i = 0; i < n; i++)
        if (strchr(followA, firstA[i][0]))
            return true;
    return false;
}

int main() {
    int n;
    printf("Enter the number of productions for A: ");
    scanf("%d", &n);

    char firstA[n][10], followS[10], followA[10];
    printf("Enter the FIRST set elements for A:\n");
    for (int i = 0; i < n; i++) {
        printf("FIRST(A%d): ", i + 1);
        scanf("%s", firstA[i]);
    }

    printf("Enter the FOLLOW set for S: ");
    scanf("%s", followS);
    printf("Enter the FOLLOW set for A: ");
    scanf("%s", followA);

    printf("\nFIRST(A): { ");
    for (int i = 0; i < n; i++)
        printf("%s%s", firstA[i], (i == n - 1) ? " " : ", ");
    printf("}\nFOLLOW(S): { %s }\nFOLLOW(A): { %s }\n", followS, followA);

    if (hasFirstFirstConflict(firstA, n))
        printf("FIRST/FIRST conflict detected! Grammar is not LL(1).\n");
    else if (hasFirstFollowConflict(firstA, n, followA))
        printf("FIRST/FOLLOW conflict detected! Grammar is not LL(1).\n");
    else
        printf("The grammar is LL(1).\n");

    return 0;
}
```

OUTPUT:

```
csc15@linux-p2-1272il:~/CS22130$ vi Practical4.c
csc15@linux-p2-1272il:~/CS22130$ cc Practical4.c
csc15@linux-p2-1272il:~/CS22130$ ./a.out
Enter the number of productions for A: 2
Enter the FIRST set elements for A:
FIRST(A1): c
FIRST(A2): ε
Enter the FOLLOW set for S: $
Enter the FOLLOW set for A: b

FIRST(A): { c, ε }
FOLLOW(S): { $ }
FOLLOW(A): { b }
The grammar is LL(1).
csc15@linux-p2-1272il:~/CS22130$ cc Practical4.c
csc15@linux-p2-1272il:~/CS22130$ ./a.out
Enter the number of productions for A: 2
Enter the FIRST set elements for A:
FIRST(A1): a
FIRST(A2): a
Enter the FOLLOW set for S: $
Enter the FOLLOW set for A: $

FIRST(A): { a, a }
FOLLOW(S): { $ }
FOLLOW(A): { $ }
FIRST/FIRST conflict detected! Grammar is not LL(1).
csc15@linux-p2-1272il:~/CS22130$ vi Practical4.c
csc15@linux-p2-1272il:~/CS22130$ vi Practical4.c
csc15@linux-p2-1272il:~/CS22130$
```

Enter the CFG:

Predictive parsing table is:

CONCLUSION:

DISCUSSION AND VIVA VOCE:

Q1: What are the application of Predictive Parser (LL(1))?
Q2: What do you mean by LL(1) Parser?
Q3: Differentiate Backtracking and Non-Backtracking Parser?
Q4: What is TOP-DOWN Parser?
Q5: Can left recursive grammar be parsed by LL(1) parser?

REFERENCES:

- Book: Compiler Design by O.G. Kakde, Laxmi Publications, 2006.
- Lab Manual of Compiler Design (Institute of Aeronautical Engineering, Dundigal, Hyderabad).