**Compiler Design(18CSC304J)**

**Experiment 10**

**SHIFT REDUCE PARSING**

Harsh Goel

RA1811003010185

**Aim:** To study and implement Shift Reduce Parser.

**Language: C++**

**Procedure:**

1. Start the program.
2. Initialize the required variables.
3. Enter the input symbol.
4. Perform the following:

for top-of-stack symbol, s, and next input symbol, a

Shift x: (x is a STATE number)

Push a, then x on the top of the stack

Advance ip to point to the next input symbol.

Reduce y: (y is a PRODUCTION number)

Assume that the production is of the form A→ß

Pop 2 \* |ß| symbols of the stack.

At this point the top of the stack should be a state number, say s’.

Push A, then goto of T[s’,A] (a state number) on the top of the stack.

Output the production A→ß.

1. Print if string is accepted or not.
2. Stop the program.

**Code Snippet:**

#include <iostream>

using namespace std;

struct prodn

{

    char p1[10];

    char p2[10];

};

void main()

{

    char input[20], stack[50], temp[50], ch[2], \*t1, \*t2, \*t;

    int i, j, s1, s2, s, count = 0;

    struct prodn p[10];

    FILE \*fp = fopen("sr\_input.txt", "r");

    stack[0] = '\0';

    printf("\n Enter the input string\n");

    scanf("%s", &input);

    while (!feof(fp))

    {

        fscanf(fp, "%s\n", temp);

        t1 = strtok(temp, "->");

        t2 = strtok(NULL, "->");

        strcpy(p[count].p1, t1);

        strcpy(p[count].p2, t2);

        count++;

    }

    i = 0;

    while (1)

    {

        if (i < strlen(input))

        {

            ch[0] = input[i];

            ch[1] = '\0';

            i++;

            strcat(stack, ch);

            printf("%s\n", stack);

        }

        for (j = 0; j < count; j++)

        {

            t = strstr(stack, p[j].p2);

            if (t != NULL)

            {

                s1 = strlen(stack);

                s2 = strlen(t);

                s = s1 - s2;

                stack[s] = '\0';

                strcat(stack, p[j].p1);

                printf("%s\n", stack);

                j = -1;

            }

        }

        if (strcmp(stack, "E") == 0 && i == strlen(input))

        {

            printf("\n Accepted");

            break;

        }

        if (i == strlen(input))

        {

            printf("\n Not Accepted");

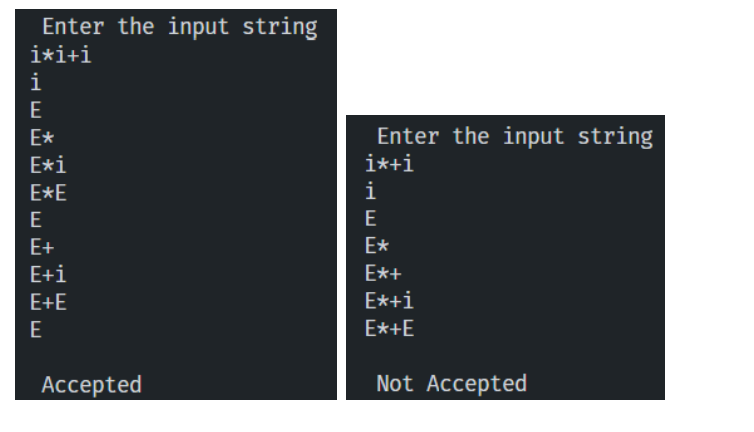
            break;

        }

    }

}

**Output Screenshots:**



**Result:**

The code was successfully implemented and output was recorded.