**CSA1443- Compiler Design for Intraprocedural Analysis**

**192321047 – KAUSHIK NARAYANAN.V**

**13)** **Write a C program to implement either Top Down parsing technique or Bottom Up Parsing technique to check whether the given input string is satisfying the grammar or not.**

**Aim:**

To implement a Bottom-Up Parsing technique (Shift-Reduce Parsing) in C to check whether the given input string satisfies the specified grammar.

**C Code Implementation:**

#include <stdio.h>

#include <string.h>

char stack[50];

char input[50];

int top = -1;

int ip = 0;

void push(char c) {

stack[++top] = c;

}

void pop() {

top--;

}

void display() {

printf("\nStack: %s\t Input: %s", stack, input + ip);

}

int reduce() {

if (top >= 2) {

if (stack[top] == 'E' && stack[top - 1] == '+' && stack[top - 2] == 'E') {

printf("\nReduce by E -> E+E");

top -= 2;

return 1;

}

if (stack[top] == 'E' && stack[top - 1] == '\*' && stack[top - 2] == 'E') {

printf("\nReduce by E -> E\*E");

top -= 2;

return 1;

}

}

if (top >= 2) {

if (stack[top] == ')' && stack[top - 1] == 'E' && stack[top - 2] == '(') {

printf("\nReduce by E -> (E)");

top -= 2;

return 1;

}

}

if (top >= 0) {

if (stack[top] == 'a') {

printf("\nReduce by E -> a");

stack[top] = 'E';

return 1;

}

}

return 0;

}

int main() {

printf("Enter the input string ending with $: ");

scanf("%s", input);

push('$');

printf("\nBottom-Up Parsing (Shift-Reduce) Simulation:\n");

display();

while (1) {

if (input[ip] != '\0') {

push(input[ip++]);

printf("\nShift '%c'", stack[top]);

display();

}

while (reduce()) {

display();

}

if (input[ip] == '\0' && top == 1 && stack[top] == 'E' && stack[0] == '$') {

printf("\n\nThe input string is successfully parsed!\n");

break;

}

if (input[ip] == '\0' && top != 1) {

printf("\n\nError: The input string cannot be parsed by the given grammar.\n");

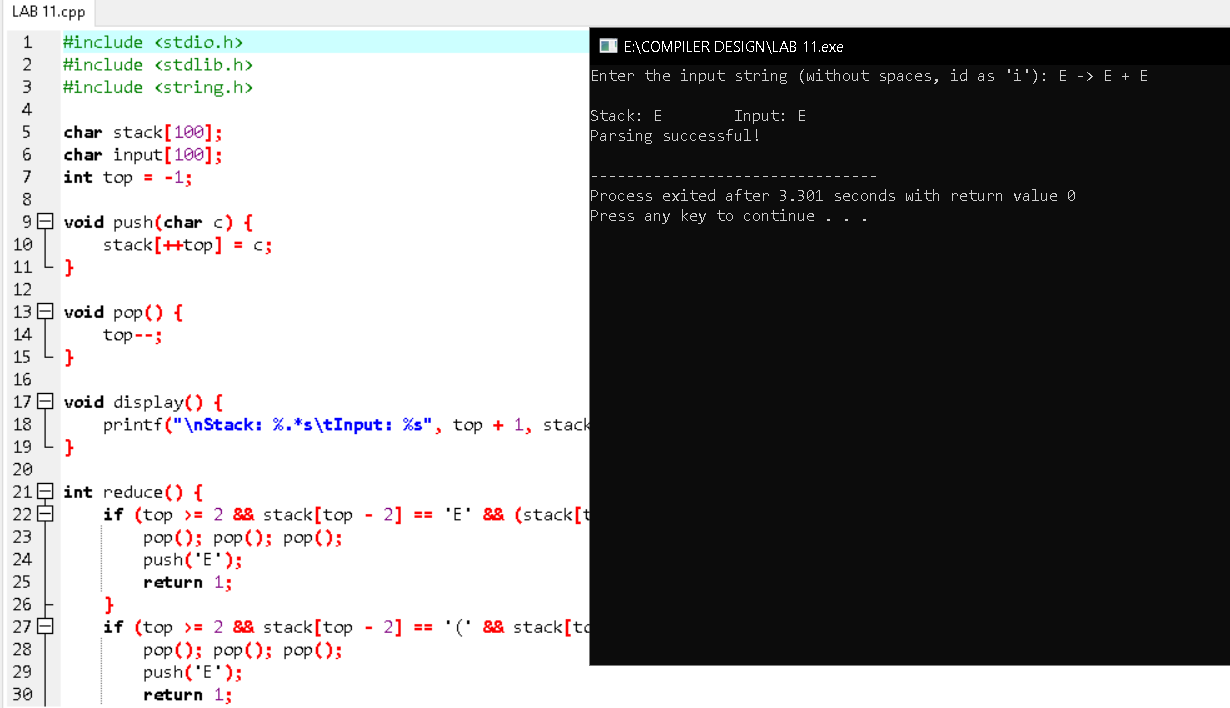
break;

}

}

return 0;

}



**14)Implement the concept of Shift reduce parsing in C Programming.**

**Aim:**To implement the Shift-Reduce Parsing technique in C to parse a given input string based on a specified grammar.

**Code:**

#include <stdio.h>

#include <string.h>

char stack[50];

char input[50];

int top = -1, ip = 0;

void push(char c) { stack[++top] = c; }

void pop() { top--; }

int reduce() {

if (top >= 2) {

if (stack[top] == 'E' && stack[top - 1] == '+' && stack[top - 2] == 'E') {

top -= 2;

stack[top] = 'E';

return 1;

}

if (stack[top] == 'E' && stack[top - 1] == '\*' && stack[top - 2] == 'E') {

top -= 2;

stack[top] = 'E';

return 1;

}

}

if (top >= 2 && stack[top] == ')' && stack[top - 1] == 'E' && stack[top - 2] == '(') {

top -= 2;

stack[top] = 'E';

return 1;

}

if (top >= 0 && stack[top] == 'a') {

stack[top] = 'E';

return 1;

}

return 0;

}

int main() {

printf("Enter input string (end with $): ");

scanf("%s", input);

push('$');

while (1) {

if (input[ip] != '\0') {

push(input[ip++]);

}

while (reduce());

if (input[ip] == '\0' && top == 1 && stack[0] == '$' && stack[1] == 'E') {

printf("String is successfully parsed!\n");

break;

}

if (input[ip] == '\0' && top != 1) {

printf("Error: Parsing failed.\n");

break;

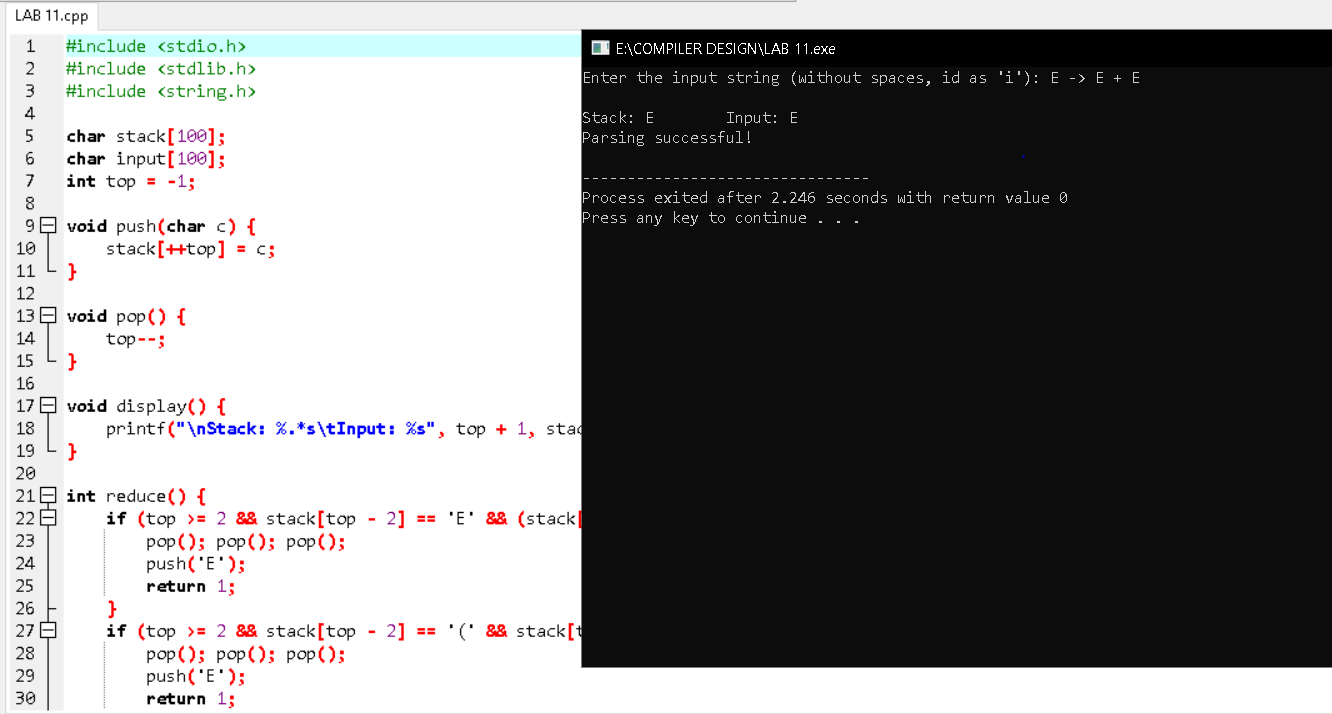
}

}

return 0;

}

**Output:**

****