LAB9

NAME: AYANIKA PAUL

REG NO. 220905128

ROLL NO. 22

D1

Q1. Develop an SLR(1) parser for the given expression grammar and demonstrate parsing actions.
E->E+T|T
T-> T*F|F
F-> (E)|id

Code:

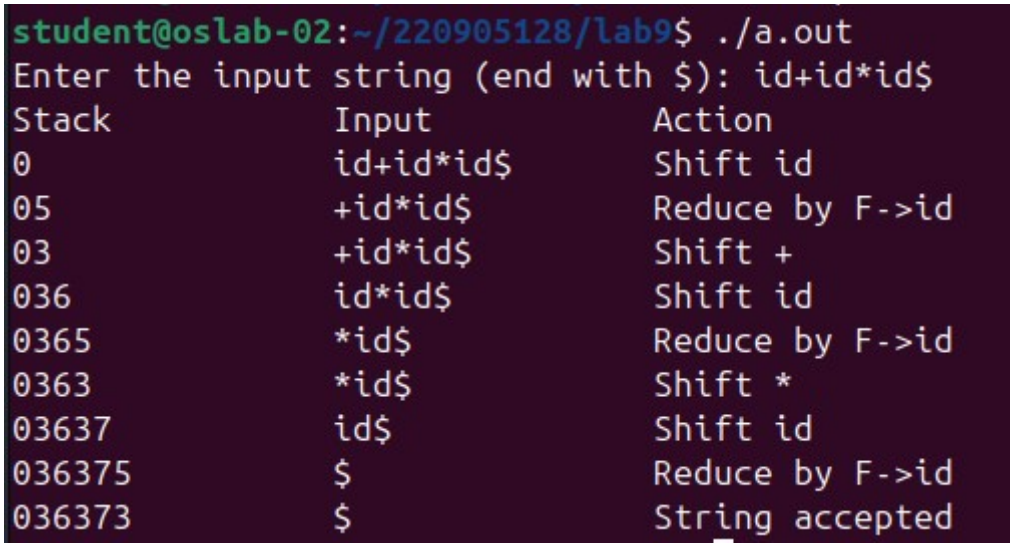
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
int ACTION[13][6] = {
    {4, 0, 0, 0, 5, 0},
    {0, 0, 6, 0, 0, 0},
    {0, 2, 2, 7, 0, 0},
    {0, 4, 4, 4, 0, 0},
    {0, 0, 0, 0, 5, 0},
    {0, 6, 6, 6, 0, 5},
    {4, 0, 0, 0, 0, 5},
    {4, 0, 0, 0, 0, 5},
    {0, 12, 6, 0, 0, 0},
    {0, 2, 2, 7, 0, 10},
    {0, 1, 1, 0, 0, 11},
    {0, 4, 4, 4, 0, 12},
    {0, 5, 5, 5, 0, 0}
};
int GOTO[13][3] = {
    {1, 2, 3},
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0},
    {8, 9, 3},
    {0, 0, 0},
    {0, 10, 3},
    {0, 0, 11},
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0}
};
int stack[MAX], top = -1;
void push(int val) {
    if (top < MAX - 1) stack[++top] = val;
}
int pop() {
    return (top >= 0) ? stack[top--] : -1;
}
int getState() {
    return stack[top];
}
void printStack() {
    for (int i = 0; i <= top; i++) {
        if (stack[i] < 10) printf("%d", stack[i]);
        else printf("%c", stack[i]);
    }
```

```
}
int symbolToIndex(char sym) {
    switch (sym) {
        case '+': return 1;
        case '*': return 2;
        case '(': return 3;
        case ')': return 4;
        case '$': return 5;
        case 'i': return 0;
        default: return -1;
    }
}
int nonTerminalToIndex(char sym) {
    switch (sym) {
        case 'E': return 0;
        case 'T': return 1;
        case 'F': return 2;
        default: return -1;
    }
}
void parse(char *input) {
    int ip = 0;
    push(0);
    printf("Stack\t\tInput\t\tAction\n");
    while (1) {
        printStack();
        printf("\t%s\t", input + ip);
        int state = getState();
        int symIndex = symbolToIndex(input[ip]);
        if (symIndex == -1) {
            printf("Error: Invalid symbol '%c'\n", input[ip]);
            return;
        }
        int action = ACTION[state][symIndex];
        if (action == 0) {
            printf("Error: Unexpected symbol '%c' at position %d\n", input[ip], ip + 1);
            return;
        } else if (action == 0) {
            printf("Accept\n");
            return;
        } else if (action > 0) {
            printf("Shift %d\n", action);
            push(input[ip]);
            push(action);
            ip++;
        } else if (action < 0) {
            int rule = -action;
            switch (rule) {
                case 1:
```

```
                case 1:
                    pop(); pop(); pop(); pop(); pop();
                    push('E');
                    push(GOTO[getState()][0]);
                    printf("Reduce by E->E+T\n");
                    break;
                case 2:
                    pop(); pop();
                    push('E');
                    push(GOTO[getState()][0]);
                    printf("Reduce by E->T\n");
                    break;
                case 3:
                    pop(); pop(); pop(); pop(); pop();
                    push('T');
                    push(GOTO[getState()][1]);
                    printf("Reduce by T->T*F\n");
                    break;
                case 4:
                    pop(); pop();
                    push('T');
                    push(GOTO[getState()][1]);
                    printf("Reduce by T->F\n");
                    break;
                case 5:
                    pop(); pop(); pop(); pop(); pop();
                    push('F');
                    push(GOTO[getState()][2]);
                    printf("Reduce by F->(E)\n");
                    break;
                case 6:
                    pop(); pop();
                    push('F');
                    push(GOTO[getState()][2]);
                    printf("Reduce by F->id\n");
                    break;
            }
        }
    }
}

int main() {
    char input[MAX];
    printf("Enter the input string (end with $): ");
    scanf("%s", input);
    parse(input);
    return 0;
}
```

OUTPUT:


student@oslab-02:~/220905128/lab9$ ./a.out
Enter the input string (end with $): id+id*id$
Stack Input Action
0 id+id*id$ Shift id
05 +id*id$ Reduce by F->id
03 +id*id$ Shift +
036 id*id$ Shift id
0365 *id$ Reduce by F->id
0363 *id$ Shift *
03637 id$ Shift id
036375 $ Reduce by F->id
036373 $ String accepted