Nithin S
221IT085

IT250 Lab Assignment 9

Q1. Implementation of shift reduce parser

Roll_No:2110491211IT064 - 2110491211IT085

Production Rules :

S → S+S

S → S*S

S → (S)

S → x

S → y

Input : 1) (x+x)*a

2) +x*y+(x)

C Code

```c
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

typedef struct Production {
    int size;
    char endChar;
    char arr[20];
} Production;

typedef struct Productions {
    int size;
    Production array[20];
    char termStart;
} Productions;

void newProductionSet(Productions *p){
    p->size = 0;
}

void addProduction(Productions *p, char endChar, char *buffer){
    int i;
    p->size++;

    for (i = 0; buffer[i] != '\0' && buffer[i] != '\n'; i++) {
        p->array[p->size - 1].arr[i] = buffer[i];
    }

    p->array[p->size - 1].size = i;
    p->array[p->size - 1].endChar = endChar;
}

void printProductions(Production *p){
    for (int i = 0; i < p->size; i++) {
        printf("%c", p->arr[i]);
    }
}

void shift(int *top, char *st, char *buffer, int *ctr, char ch){

        printf("shift\n");
        st[*top] = ch;
        st[(*top)++ + 1] = '\0';
        (*ctr)++;

        if (buffer[*ctr] == '\n' || buffer[*ctr] == '\0') ch = '$';
        else ch = buffer[*ctr];
}

void reduce(Productions *set, char *buffer, char *st, int *top, int *ctr) {
    int prod;
    int flag = 1;
```

```
    while (flag) {
        for (prod = 0; prod < set->size; prod++) {
            flag = 1;
            int k = set->array[prod].size - 1;

            for (int j = *top - 1; j >= 0 && k >= 0; j--, k--) {
                if (set->array[prod].arr[k] != st[j]) {
                    flag = 0;
                    break;
                }
            }
            if (k != -1) flag = 0;
            if (flag) break;
        }

        if (flag) {
            st[*top] = '\0';
            printf("%s\t\t\t%s\t\t\t", st, buffer + *ctr);
            printf("reduce by ");
            printf(" %c -> ", set->array[prod].endChar);
            printProductions(&set->array[prod]);
            printf("\n");

            for (int i = 0; i < set->array[prod].size; i++) {
                (*top)--;
            }

            st[*top] = set->array[prod].endChar;
            st[*top + 1] = '\0';
            (*top)++;

        } else break;
    }
}


bool shiftReduceParser(Productions *set, char *buffer){

    char st[1000];
    int top = 0, ctr = 0;
    char ch;

    st[top++] = '$';

    printf("Stack\t\t\tInput\t\t\tAction\n\n");

    while (true) {

        if (buffer[ctr] == '\n' || buffer[ctr] == '\0') ch = '$';
        else ch = buffer[ctr];
        st[top] = '\0';

        printf("%s\t\t\t%s\t\t\t", st, buffer + ctr);
```

```
        else ch = buffer[ctr];
        st[top] = '\0';

        printf("%s\t\t\t%s\t\t\t", st, buffer + ctr);

        if (ch == '$'){
            if(top == 2 && st[1] == set->termStart) {
                printf("accepted\n");
                return true;
            }
            else if (top != 1 || st[0] != set->termStart) {
                printf("reject\n");
                return false;
            }
            else {
                printf("accepted\n");
                return true;
            }
        }

        shift(&top, st, buffer, &ctr, ch);
        //If possible, then reduce
        reduce(set, buffer, st, &top, &ctr);
    }
    return false;
}


void removeSpaces(char* str){
    int i, j;
    for (i = 0, j = 0; str[i] != '\0'; i++)
    {
        if (!isspace(str[i]))
            str[j++] = str[i];
    }
    str[j] = '\0';
}


int main()
{
    char ch;
    char temp[1000], show[1000];

    // Inserting Productions given in the question
    Productions g;
    newProductionSet(&g);

    char prod1[4] = {'S','+','S','\0'};
    char prod2[4] = {'S','*','S','\0'};
    char prod3[4] = {'(','S',')','\0'};
    char prod4[2] = {'x','\0'};
    char prod5[2] = {'y','\0'};
```

```
    char temp[1000], show[1000];

    // Inserting Productions given in the question
    Productions g;
    newProductionSet(&g);

    char prod1[4] = {'S','+','S','\0'};
    char prod2[4] = {'S','*','S','\0'};
    char prod3[4] = {'(','S',')','\0'};
    char prod4[2] = {'x','\0'};
    char prod5[2] = {'y','\0'};

    addProduction(&g, 'S', prod1);
    addProduction(&g, 'S', prod2);
    addProduction(&g, 'S', prod3);
    addProduction(&g, 'S', prod4);
    addProduction(&g, 'S', prod5);

    g.termStart = 'S';

    printf("\n");

    printf("Enter the string to parse: ");

    scanf(" %[^\n]", temp);

    strcpy(show, temp);

    removeSpaces(temp);

    int len = strlen(temp);

    if (len > 0 && temp[len - 1] != '$') {
        temp[len] = '$';
        temp[len + 1] = '\0';
    } else {
    strcat(temp, "$");
    }

    printf("\nGrammar Productions:\n\n");

    for (int i = 0; i < g.size; i++) {
        printf("%c -> ", g.array[i].endChar);
        printProductions(&g.array[i]);
        printf("\n");
    }

    printf("\n\n");

    if (shiftReduceParser(&g, temp)) printf("\nString %s is accepted\n", show);
    else printf("\nString %s is rejected\n\n", show);

    return 0;
}
```

Output

Test Case 1 : (x+x)*a

```
student@HP-Elite600G9-08:~/Desktop/assgn$ gcc parser.c
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the string to parse: (x+x)*a

Grammar Productions:

S -> S+S
S -> S*S
S -> (S)
S -> x
S -> y


Stack                   Input                   Action

$                       (x+x)*a$                        shift
$(                      x+x)*a$                 shift
$(x                     +x)*a$                  reduce by  S -> x
$(S                     +x)*a$                  shift
$(S+                    x)*a$                   shift
$(S+x                   )*a$                    reduce by  S -> x
$(S+S                   )*a$                    reduce by  S -> S+S
$(S                     )*a$                    shift
$(S)                    *a$                     reduce by  S -> (S)
$S                      *a$                     shift
$S*                     a$                      shift
$S*a                    $                       reject

String (x+x)*a is rejected
```

Test Case 2 : +x*y+(x)

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the string to parse: +x*y+(x)

Grammar Productions:

S -> S+S
S -> S*S
S -> (S)
S -> x
S -> y


Stack                   Input                   Action

$                       +x*y+(x)$                       shift
$+                      x*y+(x)$                        shift
$+x                     *y+(x)$                 reduce by  S -> x
$+S                     *y+(x)$                 shift
$+S*                    y+(x)$                  shift
$+S*y                   +(x)$                   reduce by  S -> y
$+S*S                   +(x)$                   reduce by  S -> S*S
$+S                     +(x)$                   shift
$+S+                    (x)$                    shift
$+S+(                   x)$                     shift
$+S+(x                  )$                      reduce by  S -> x
$+S+(S                  )$                      shift
$+S+(S)                 $                       reduce by  S -> (S)
$+S+S                   $                       reduce by  S -> S+S
$+S                     $                       reject

String +x*y+(x) is rejected
```