Shift reduce parser

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX_RULES 10
#define MAX_PROD 10
#define MAX_LEN 50
char stk[50],input[50];
int top=-1;
typedef struct{

  char lhs;
  char rhs[MAX_PROD][MAX_LEN];
  int prodcount;


}Rule;

Rule grammar[MAX_RULES];
int rulecount=0;

void pushstring(const char *s){
   for(int i=0;s[i]!='\0';i++)
     stk[++top]=s[i];
   stk[top+1]='\0';
}
void pop(int n){
     top=top-n;
     stk[top+1]='\0';
}
void trim(char *s){

    int len=strlen(s);
    while(len>0 && (s[len-1]==' '||s[len-1]=='\n'))
        s[--len]='\0';
    int start=0;
    while(s[start]==' ')
        start++;
    if(start>0)memmove(s,s+start,strlen(s+start)+1);

}
int checkreduce(){

 for(int r=0;r<rulecount;r++)
 {
       for(int p=0;p<grammar[r].prodcount;p++)
       {
           int len=strlen(grammar[r].rhs[p]);
```

```
            if(len<=0)continue;
            if(top+1>=len){
                if(strncmp(&stk[top-len+1],grammar[r].rhs[p],len)==0)
                {
                   pop(len);
                   pushstring((char[]){grammar[r].lhs,'\0'});
                   printf("\n$%s\t%s$\tREDUCE TO
%c->%s",stk,input,grammar[r].lhs,grammar[r].rhs[p]);
                    return 1;
                }

            }
        }
     }
     return 0;
}
int main(){
   printf("enter the no of grammar rules ");
   scanf("%d",&rulecount);
   getchar();
   for(int i=0;i<rulecount;i++)
   {
       char line[200];
       printf("Enter LHS non-terminal:");
       scanf("%c",&grammar[i].lhs);
       getchar();
       printf("Enter productions for  %c: ",grammar[i].lhs);
       fgets(line,sizeof(line),stdin);
       trim(line);
       char *token=strtok(line,",");
       int p=0;
       while(token!=NULL){

          trim(token);
          strcpy(grammar[i].rhs[p++],token);
          token=strtok(NULL,",");

      }
       grammar[i].prodcount=p;
   }
   printf("Enter input string:");
   scanf("%s",input);
   printf("\nstack\tinput\taction\n");
   int len=strlen(input);
   for(int i=0;i<len;i++){
      pushstring((char[]){input[i],'\0'});
      input[i]=' ';
      printf("\n$%s\t%s$\tSHIFT->%c",stk,input,stk[top]);
```

```
    while(checkreduce());
  }
  while(checkreduce());
  if(top==0 && stk[0]==grammar[0].lhs)
    printf("\n\nInput string successfully parsed!!\n");
  else
    printf("\n\nsyntax error ! parsing failed\n");
  return 0;
}
```

Output

```
enter the no of grammar rules 1
Enter LHS non-terminal:E
Enter productions for  E: E+E,E*E,(E),a
Enter input string:a+a*a

stack    input    action

$a     +a*a$    SHIFT->a
$E     +a*a$    REDUCE TO E->a
$E+      a*a$    SHIFT->+
$E+a       *a$    SHIFT->a
$E+E       *a$    REDUCE TO E->a
$E       *a$    REDUCE TO E->E+E
$E*        a$    SHIFT->*
$E*a         $    SHIFT->a
$E*E         $    REDUCE TO E->a
$E         $    REDUCE TO E->E*E

Input string successfully parsed!!
```

Recursive descent parsing

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char input[20];
int i=0,error=0;
void E();
void T();
void Eprime();
void Tprime();
void F();
void main(){

  printf("Enter an arithmetic expression: \n");
  gets(input);
  E();
  if(strlen(input)==i && error==0)
  printf("Accepted....!!\n");
  else
  printf("Rejected....!!\n");

}
void E(){
 T();
 Eprime();
}
void Eprime(){
 if(input[i]=='+')
 {
   i++;
   T();
   Eprime();
 }
}
void T(){
 F();
 Tprime();
}
void Tprime(){
 if(input[i]=='*'){
     i++;
     F();
     Tprime();
 }
}
void F(){
 if(isalnum(input[i]))
  i++;
```

```
 else if(input[i]=='('){
     i++;
     E();
     if(input[i]==')')
       i++;
     else
       error=1;
 }
 else
  error=1;
}
```

Output

```
Enter an arithmetic expression:
a+b*c
Accepted....!!

Enter an arithmetic expression:
a++b
Rejected....!!
```