EX. NO. 7
19/03/21

SHIFT REDUCE PARSER

AIM: To construct a Shift reducing parser parser

ALGORITHM:

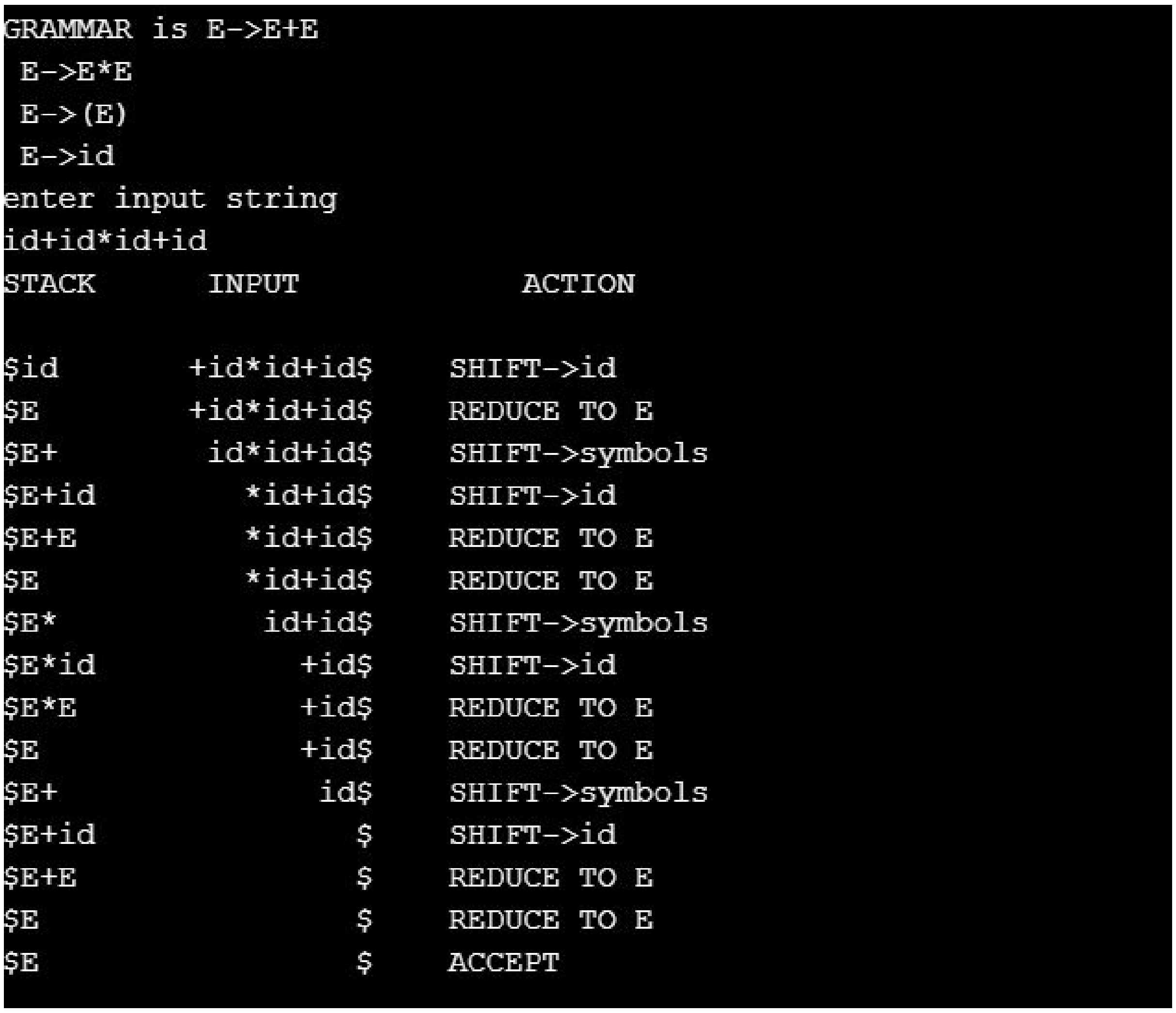
```
1.Start the program
2.read the variables , stack symbols
3.loop forever: for top-of-stack symbol, s, and next input symbol, a case action of T[s,a]
4.shift x:(x is a STATE number) push a, then x on the top of the stack and advance ip to the next input symbol.
5. reduce y: (y is a PRODUCTION number) Assume that the production is of the form
   A ==> beta
6. pop 2 * |beta| 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
7.Output the production A ==> beta
8.accept: return --- a successful parse.
9.default: error --- the input string is not in the language.
10.Stop the program
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main()
  {

    puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
    puts("enter input string ");
    gets(a);
    c=strlen(a);
    strcpy(act,"SHIFT->");
    puts("STACK \t   INPUT \t    ACTION");


    for(k=0,i=0; j<c; k++,i++,j++) //loop for entire input string
     {
      if(a[j]=='i' && a[j+1]=='d') //if input is 'id'
        {
          stk[i]=a[j];
          stk[i+1]=a[j+1];
          stk[i+2]='\0';
          a[j]=' ';
```

```
            a[j+1]=' ';
            printf("\n$%s\t%s$\t%sid",stk,a,act); //shift
            check(); //check for reduction
          }
        else //if input is operator
          {
            stk[i]=a[j];
            stk[i+1]='\0';
            a[j]=' ';
            printf("\n$%s\t%s$\t%ssymbols",stk,a,act); //shift
            check(); //check for reduction
          }
       }
    if(stk[0] == 'E' && stk[1] == '\0')
        printf("\n$%s\t%s$\tACCEPT",stk,a,act);
    else //else reject
        printf("\n$%s\t%s$\tERROR",stk,a,act);
   }

 //check for reduction
void check()
  {
    strcpy(ac,"REDUCE TO E");
    if(stk[0] == 'E' && stk[1] == '\0' && a[0] == '$' && a[1] == '\0')
    {
        printf("hello");
    }
    for(z=0; z<c; z++)
      if(stk[z]=='i' && stk[z+1]=='d') //production 4
        {
          stk[z]='E';
          stk[z+1]='\0';
          printf("\n$%s\t%s$\t%s",stk,a,ac);
          j++;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E') //production 1
        {
          stk[z]='E';
          stk[z+1]='\0';
          stk[z+2]='\0';
          printf("\n$%s\t%s$\t%s",stk,a,ac);
          i=i-2;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E') //production 2
        {
          stk[z]='E';
          stk[z+1]='\0';
          stk[z+1]='\0';
          printf("\n$%s\t%s$\t%s",stk,a,ac);
          i=i-2;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')') //production 3
        {
          stk[z]='E';
          stk[z+1]='\0';
          stk[z+1]='\0';
          printf("\n$%s\t%s$\t%s",stk,a,ac);
          i=i-2;
        }
  }
```

OUTPUT:



```
GRAMMAR is E->E+E
 E->E*E
 E->(E)
 E->id
enter input string
id+id*id+id
STACK      INPUT           ACTION

$id       +id*id+id$    SHIFT->id
$E        +id*id+id$    REDUCE TO E
$E+        id*id+id$    SHIFT->symbols
$E+id        *id+id$    SHIFT->id
$E+E         *id+id$    REDUCE TO E
$E           *id+id$    REDUCE TO E
$E*           id+id$    SHIFT->symbols
$E*id           +id$    SHIFT->id
$E*E            +id$    REDUCE TO E
$E              +id$    REDUCE TO E
$E+              id$    SHIFT->symbols
$E+id              $    SHIFT->id
$E+E               $    REDUCE TO E
$E                 $    REDUCE TO E
$E                 $    ACCEPT
```

RESULT: Shift reduce parser was successfully implemented using C.