

Compiler Design Lab

Recursive Decent Parser

1. Grammar -

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \epsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \epsilon \\ F &\rightarrow (E) \mid i \end{aligned}$$

Program -

```
#include<stdio.h>
#include<string.h>
int E(),Edash(),T(),Tdash(),F();
char *ip;
char string[50];
int main()
{
printf("Enter the string\n");
scanf("%s",string);
ip=string;
printf("\n\nInput\tAction\n-----\n");

if(E() && *ip=='\0'){
printf("\n-----\n");
printf("\n String is successfully parsed\n");
}
else{
printf("\n-----\n");
printf("Error in parsing String\n");
}
}
int E()
{
```

```

printf("%s\tE->TE' \n",ip);
if(T())
{
if(Edash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
int Edash()
{
if(*ip=='+')
{
printf("%s\tE' ->+TE'
\n",ip); ip++;
if(T())
{
if(Edash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tE' ->^
\n",ip); return 1;
}
}

```

```

}
int T()
{
printf("%s\tT->FT'
\n",ip); if(F())
{

if(Tdash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
int Tdash()
{
if(*ip=='*')
{
printf("%s\tT'->*FT'
\n",ip); ip++;
if(F())
{
if(Tdash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else

```

```

{
printf("%s\tT'->^
\n",ip); return 1;
}
}
int F()
{
if(*ip=='(')
{
printf("%s\tF->(E)
\n",ip); ip++;
if(E())
{
if(*ip==')')
{
ip++;
return 0;
}
else
return 0;
}
else
return 0;
}

else if(*ip=='i')
{
ip++;
printf("%s\tF->id \n",ip);
return 1;
}
else
return 0;
}

```

Test Cases -

i+i*i String is successfully parsed
i+i String is successfully parsed
i*i String is successfully parsed
i*i+i*i+i String is successfully parsed
i+*+i Error in parsing String
i+i* Error in parsing String

2. Grammar -

$S \rightarrow (L) \mid a$

$L \rightarrow ST$

$T \rightarrow ,ST \mid \diamond\diamond$

Program -

```
#include<stdio.h>
```

```
#include<string.h>
```

```
int S(),L(),Ldash();
```

```
char *ip;
```

```
char string[50];
```

```
int main()
```

```

{

printf("Enter the string:\n");

scanf("%s",string);

ip=string;

printf("\n\nInput\tAction\n-----\n");

if(S() && *ip=='\0'){

    printf("\n-----\n");

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

}

else{

    printf("\n-----\n");

    printf("Error in parsing String\n");

}

}

int S()

{

    if(*ip=='(')

    {

```

```
printf("%s\tS->(L) \n",ip);

ip++;

if(L())

{

    if(*ip=='')

    {

        ip++;

        return 1;

    }

    else

        return 0;

}

else

    return 0;

}

else if(*ip=='a')

{

    ip++;
```

```

        printf("%s\tS->a \n",ip);

        return 1;

    }

    else

        return 0;

}

int L()

{

    printf("%s\tL->SL' \n",ip);

    if(S())

    {

        if(Ldash())

        {

            return 1;

        }

        else

            return 0;

    }

```



```
    else

        return 0;

}

int Ldash()

{

    if(*ip==',')

    {

        printf("%s\tL' -> ,SL' \n",ip);

        ip++;

        if(S())

        {

            if(Ldash())

            {

                return 1;

            }

            else

                return 0;

        }

    }
```

```

        else

            return 0;

    }

    else

    {

        printf("%s\tL' ->? \n",ip);

        return 1;

    }

}

```

Test Cases -

(a,(a,a)) String is parsed successfully
(a,((a,a),(a,a))) String is parsed successfully
(a,a)) Error in parsing
(a,(a,a))) Error in parsing