PRACTICAL-5

AIM:

Implement a Syntax Tree.

IMPLEMENTATION:

- gcc <newly created .c file> -o <file name for exe file>
- <filename of exe file>

In this case, create a syntax.txt file as input for the executable which will contain following statements.

t1=a+b

t2=c-d

t3=e+t2

t4=t1-t3

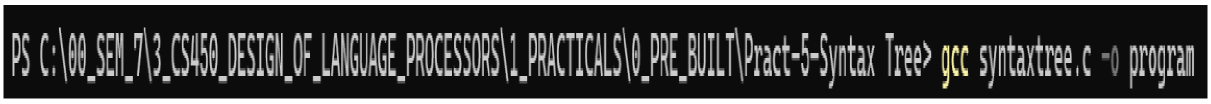
PROGRAM CODE:

```
#include<conio.h>
#include<stdio.h>
int main()
{
FILE *fp;
int i=0,j=0,k,l,row,col,s,x;
char a[10][10],ch,main[50],search;
//clrscr();
fp=fopen("syntax.txt","r+");
while((ch=fgetc(fp))!=EOF)
{
if(ch=='\n')
{
row=i;
col=j;
j=0;
i++;
}
else
{
a[i][j]=ch
; j++;
}
}
printf("\n");
for(k=0;k<row+1;k++)
{
for(l=0;l<col;l++)
{
printf("%c",a[k][l]);
}
printf("\n");
}
i=0;
s=0;
for(k=0;k<row+1;k++)
{
main[i]=a[k][1];
i++;
if(a[k][3]=='t')
{
search=a[k][4];
for(l=0;l<i;l++)
{
if(main[l]==search)
{
main[i]=main[l];
i++;
break;
}
}
main[i]=a[k][5];
s=5;
i++;
}
else
```

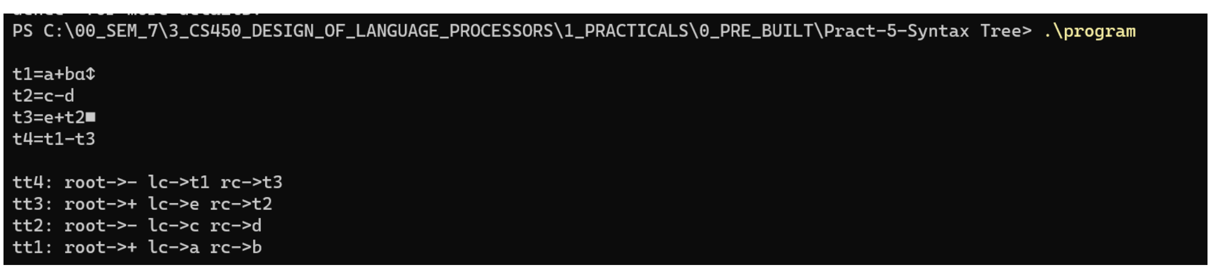
```
{
main[i]=a[k][3];
// printf("\n%c",main[i]);
i++;
main[i]=a[k][4];
// printf(",%c\n",main[i]);
s=4;
i++;
}
s++;
if(a[k][s]=='t')
{
s++;
search=a[k][s];
for(l=0;l<i;l++)
{
if(main[l]==search)
{
main[i]=main[l];
i++;
break;
}
}
}
else
{
main[i]=a[k][s];
i++;
}
}
for(x=i-1;x>=0;x=x-4)
{
printf("\ntt%c: root->%c ",main[x-3],main[x-1]);
if(main[x-2]>48 &&main[x-2]<59)
printf("lc->t%c ",main[x-2]);
else
printf("lc->%c ",main[x-2]);
if(main[x]>48 &&main[x]<59)
printf("rc->t%c ",main[x]);
else
printf("rc->%c ",main[x]);
}
```

```
getch();
}
```

OUTPUT:



```
PS C:\00_SEM_7\3_CS450_DESIGN_OF_LANGUAGE_PROCESSORS\1_PRACTICALS\0_PRE_BUILT\Pract-5-Syntax Tree> gcc syntaxtree.c -o program
```



```
PS C:\00_SEM_7\3_CS450_DESIGN_OF_LANGUAGE_PROCESSORS\1_PRACTICALS\0_PRE_BUILT\Pract-5-Syntax Tree> .\program

t1=a+bα↕
t2=c-d
t3=e+t2■
t4=t1-t3

tt4: root->- lc->t1 rc->t3
tt3: root->+ lc->e rc->t2
tt2: root->- lc->c rc->d
tt1: root->+ lc->a rc->b
```