T. LAKSHMAN
AP19110010385

CODE:

```
#include<stdio.h>

#include<ctype.h>

#include<string.h>

char keyword [30][30]={"int","while","break","for","do","if","float","char","switch","d
ouble","short","long","unsigned","sizeof","else","register","extern","static","auto
","case","break","volatile","Enum","typedef"};

char id[20], num[10];

//declare symbol table as a doubly dimensional array of characters.

char symb_tab[30][30]={""};



int check_keyword(char s[])

 {

 int i;

 for(i=0;i<24;i++)

 if(strcmp(s,keyword[i])==0)

 return 1;

 return 0;

 }



/*write a function to store identifier in symbol table


*/

int store_symb_tab(char id[])

{

//Check whether the id is already available in the symbol table, if available,  ignore. otherwise add it.

int i;

for(i=0;i<30;i++){

if(strcmp(id,symb_tab[i])==0){

return 1;
```

```
}
else{
for(i=0;i<30;i++){
if(strcmp("",symb_tab[i])==0){
strcpy(symb_tab[i], id);
return 0;
}
}
}
}
}
```

```
int main()
{
 FILE *fp1,*fp2;
 char c;
 char idx;

 int state=0;
 int i=0,j=0;
 fp1=fopen("x.txt","r");//input file containing src prog
 fp2=fopen("y.txt","w");//output file name
 while((c=fgetc(fp1))!=EOF)
 {
 switch(state)
 {
 case 0: if(isalpha(c)){
 state=1; id[i++]=c;}
 else if(isdigit(c)){
```

```
state=3; num[j++]=c;}
else if(c=='<' || c=='>'){

if (c=='<'){
idx = '<';
}
else{
idx = '>';
}

state=5;
}
else if(c=='=' || c=='!') {

if (c=='='){
idx = '=';
}
else{
idx = '!';
}

state=8;

}
else if(c=='/')
state=10;
else if(c==' ' || c=='\t' || c=='\n')  state=0;
else
fprintf(fp2,"\n%c",c);  break;
case 1:if(isalnum(c)){
state=1; id[i++]=c;
```

```
}
else{
id[i]='\0';
if(check_keyword(id)){
fprintf(fp2," \n %s : keyword ",id);
}
else{
fprintf(fp2,"\n %s : identifier",id);
store_symb_tab(id);
}
// call a function which stores id in symbol table


state=0;
i=0;
ungetc(c,fp1);
}
break;
case 3:if(isdigit(c)){
num[j++]=c;
state=3;
}
else{
num[j]='\0';
fprintf(fp2," \n%s: number",num);
state=0;
j=0;
ungetc(c,fp1);
}
break;
case 5:if(c=='='){
```

```
//fprintf(fp2,"\n relational operator ");
//write code to print specific operator like <= or >=  state=0;
fprintf(fp2,"\n%c%c relational operator ",idx,c);



}
else{
//fprintf(fp2,"\n relational operator ");
//write code to print specific operator like <, >, <= or >=
fprintf(fp2,"\n%c relational operator ",idx);

state=0;
ungetc(c,fp1);
}
break;
case 8:if(c=='='){
//fprintf(fp2,"\n relational operator ");
//write code to print specific operator like == or !=  state=0;
fprintf(fp2,"\n%c%c relational operator ",idx,c);



}
else{
ungetc(c,fp1);
state=0;
}
break;
case 10:if(c=='*')
state=11;
else
```

```
fprintf(fp2,"\n invalid lexeme");
break;
case 11: if(c=='*')
state=12;
else
state=11;
break;
case 12:if(c=='*')
state=12;
else if(c=='/')
state=0;
else
state=11;
break;
}//End of switch
}//end of while

for(int i=0;i<20;i++){
if(strcmp("",symb_tab[i])==0){
break;
}
else{
printf("%d:%s\n",i,symb_tab[i]);
}
}

 if(state==11)
 fprintf(fp2,"comment did not close");
 fclose(fp1);
 fclose(fp2);
 return 0;
```

```
}
```