**CD Practical 1:**

**Design a lexical analyser for given language and the lexical analyser should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language.**

#include<string.h>

#include<ctype.h>

#include<stdio.h>

void keyword(char str[10])

{

if(strcmp("for",str)==0||strcmp("while",str)==0||strcmp("do",str)==0||strcmp("int",str)==0||strcmp("float",str)==0||strcmp("char",str)==0||strcmp("double",str)==0||strcmp("static",str)==0||strcmp("switch",str)==0||strcmp("case",str)==0)

printf("\n %s is a keyword",str);

else

printf("\n %s is an identifier",str);

}

main()

{

FILE \*f1,\*f2,\*f3;

char c,str[10],st1[10];

int num[100],lineno=0,tokenvalue=0,i=0,j=0,k=0;

f1=fopen("prac1.txt","r");

f2=fopen("identifier","w");

f3=fopen("operator","w");

while((c=getc(f1))!=EOF) {

if(isdigit(c))

{

tokenvalue=c-'0';

c=getc(f1);

while(isdigit(c)) {

tokenvalue\*=10+c-'0';

c=getc(f1);

}

num[i++]=tokenvalue;

ungetc(c,f1);

}

else if(isalpha(c))

{

putc(c,f2);

c=getc(f1);

while(isdigit(c)||isalpha(c)||c=='\_'||c=='$')

{

putc(c,f2);

c=getc(f1);

}

putc(' ',f2);

ungetc(c,f1);

}

else if(c==' '||c=='\t')

printf(" ");

else if(c=='\n')

lineno++;

else if(c=='+'||c=='-'||c=='='||c=='/'||c=='\*'||c=='%'||c=='<'||c=='>'||c=='<='||c=='='||c=='>='||c=='<<'||c=='>>'||c=='++'||c=='--')

putc(c,f3);

else

printf(" ");

}

fclose(f2);

fclose(f3);

fclose(f1);

printf("\nThe no's in the program are");

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

printf(" %d \t",num[j]);

printf("\n");

f2=fopen("identifier","r");

k=0;

printf("The keywords and identifiers are:");

while((c=getc(f2))!=EOF) {

if(c!=' ')

str[k++]=c;

else

{

str[k]='\0';

keyword(str);

k=0; }

}

fclose(f2);

f3=fopen("operator","r");

printf("\nOperators are");

while((c=getc(f3))!=EOF)

printf(" %c \t ",c);

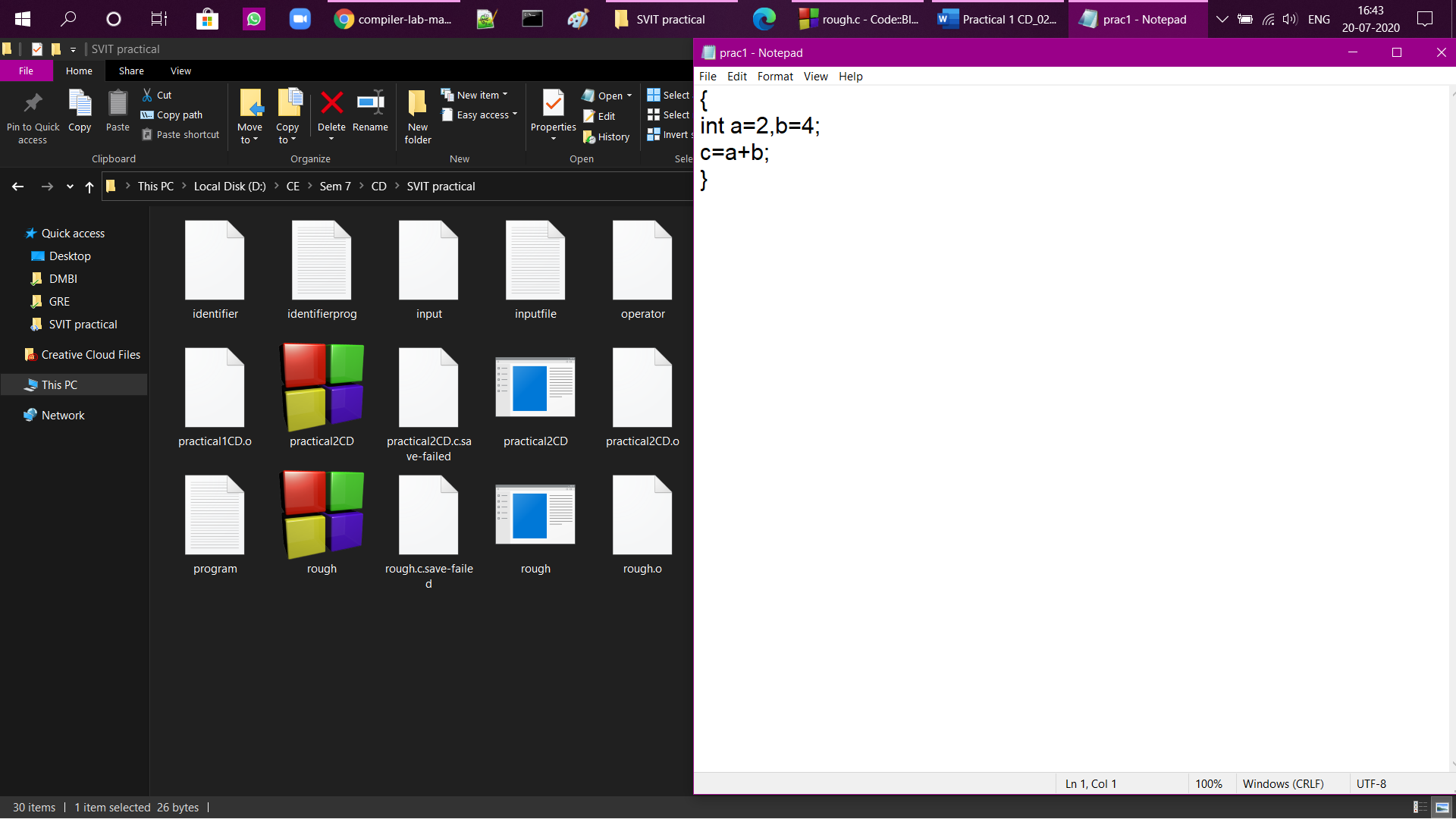
printf("\n");

fclose(f3);

printf("Total no. of lines are:%d",lineno+1);

}

**File:**



**Program output:**

