**COMPILER DESIGN LAB**

**PCS-601**

**NAME-ANURAG KUMAR**

**SECTION-G**

**CLASS ROLL NO.-09**

**UNIVERSITY ROLL NO.-2014581**

**PROGRAM:01**

**Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.**

**CODE:**

%{

int m, n, t, c;

%}

%%

\n n++;

\t m++;

[ ] t++;

. c++;

%%

int main()

{

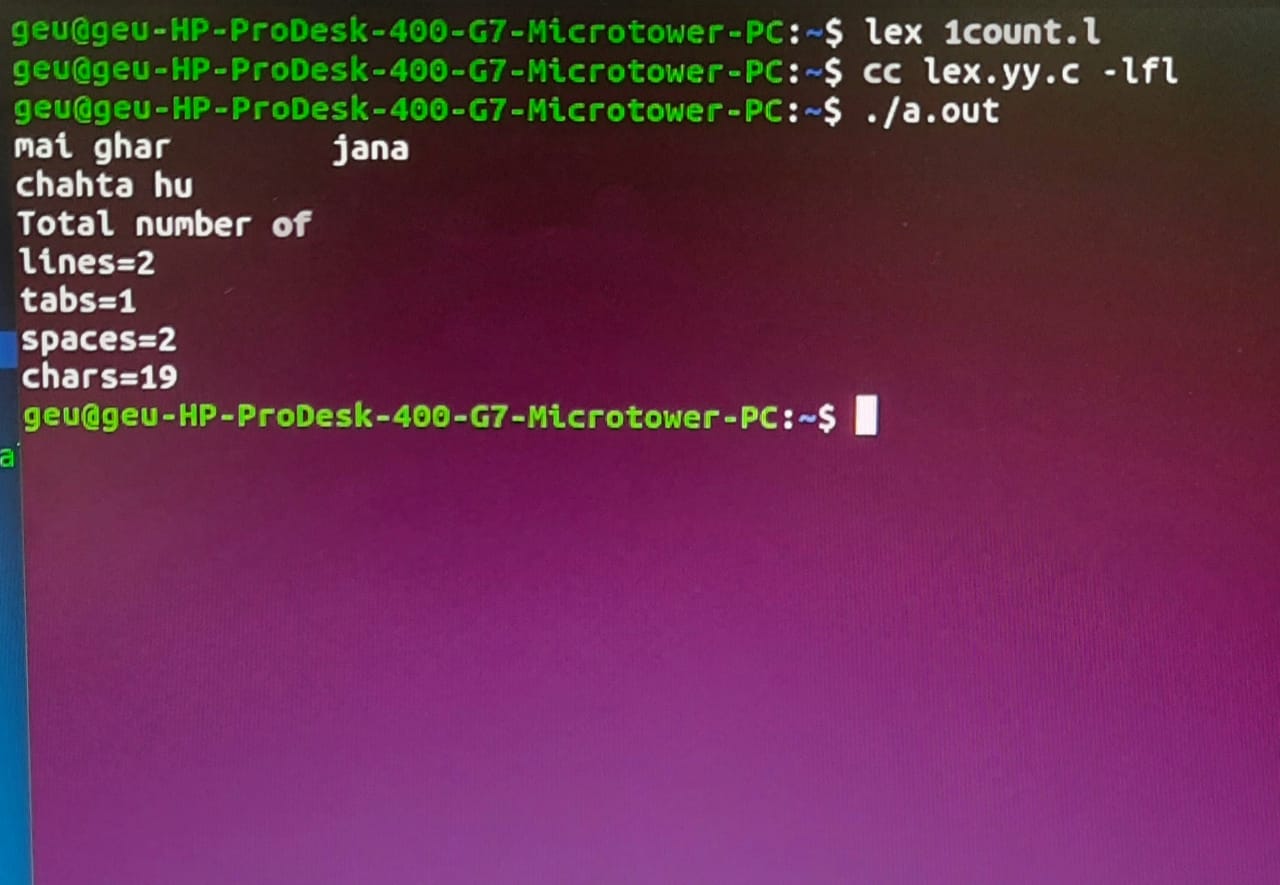
yylex();

printf("Total number of \nlines=%d \ntabs=%d \nspaces=%d \nchars=%d \n",n,m,t,c);

return 0;

}

**OUTPUT:**

****

**PROGRAN-02**

**Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.**

**CODE:**

%{

int c = 0;

%}

%%

[a-zA-Z\_][a-zA-Z0-9\_]\* {c++; printf("\t Valid Identifier = %s", yytext);}

. ;

%%

int main()

{

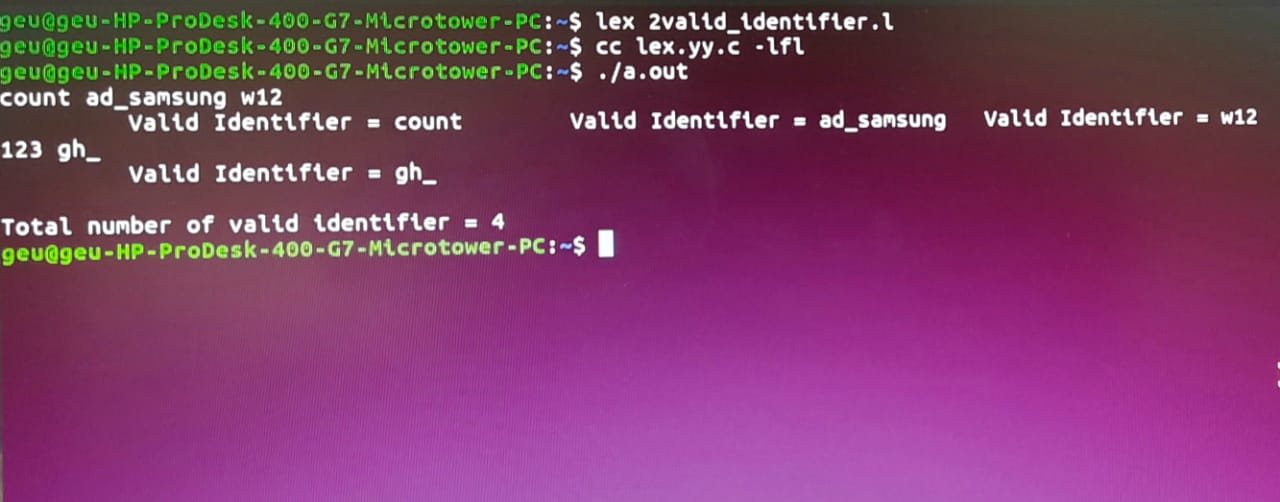
yylex();

printf("\nTotal number of valid identifier = %d\n",c);

return 0;

}

**OUTPUT:**

****

**PROGRAM-03**

**Design a LEX Code to identify and print integer and float value in given Input pattern.**

**CODE:**

%{

int m=0, n=0;

%}

%%

[0-9]+ {m++; printf("\t Integer = %s", yytext);}

[0-9]\*"."[0-9]+ {n++; printf("\t Float = %s", yytext);}

. ;

%%

int main()

{

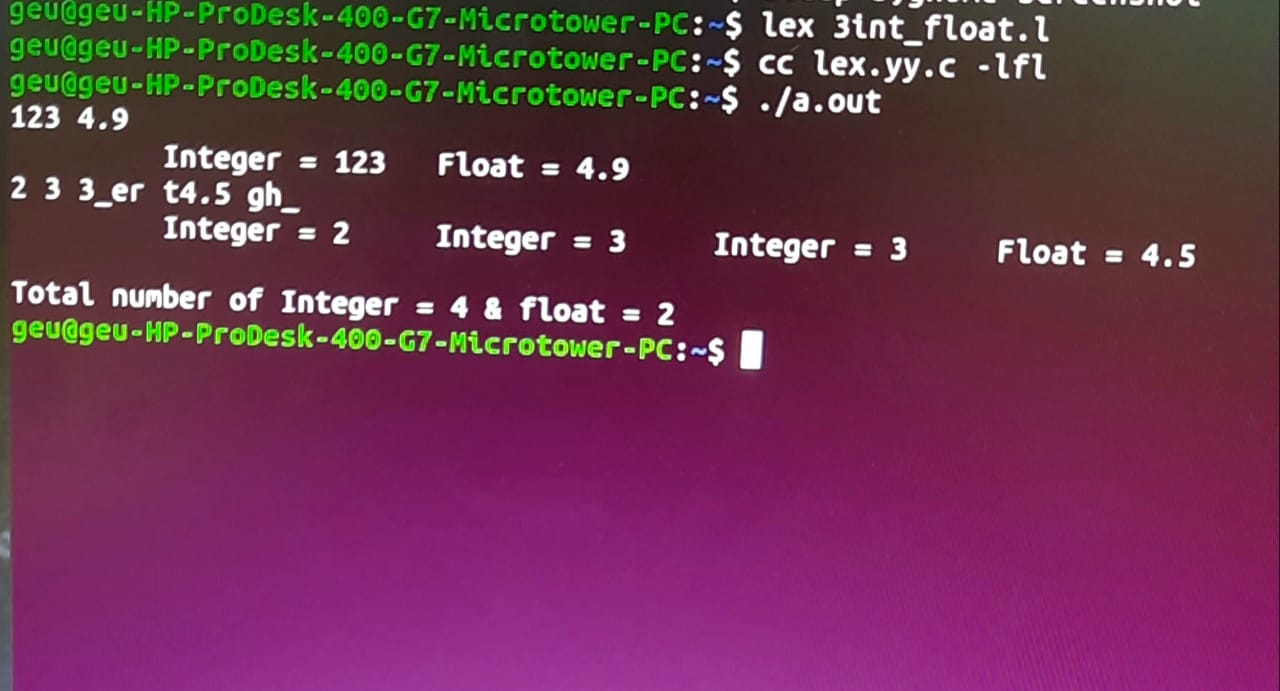
yylex();

printf("\nTotal number of Integer = %d & float = %d \n",m,n);

return 0;

}

**OUTPUT:**

****

**PROGRAM-04**

**Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS) the following C-fragment:**

**CODE:**

%{

int n = 0;

%}

%%

"while"|"if"|"else" {n++; printf("\t keywords: %s", yytext);}

"int"|"float" {n++; printf("\t keywords: %s", yytext);}

[a-zA-Z\_][a-zA-Z0-9\_]\* {n++; printf("\t Identifier: %s", yytext);}

"<="|"=="|"="|"++"|"-"|"\*"|"+" {n++; printf("\t operator: %s", yytext);}

"("|")"|"{"|"}"|","|";" {n++; printf("\t Separator: %s", yytext);}

[0-9]\*"."[0-9]+ {n++; printf("\t Float: %s", yytext);}

[0-9]+ {n++; printf("\t Integer: %s", yytext);}

. ;

%%

int main()

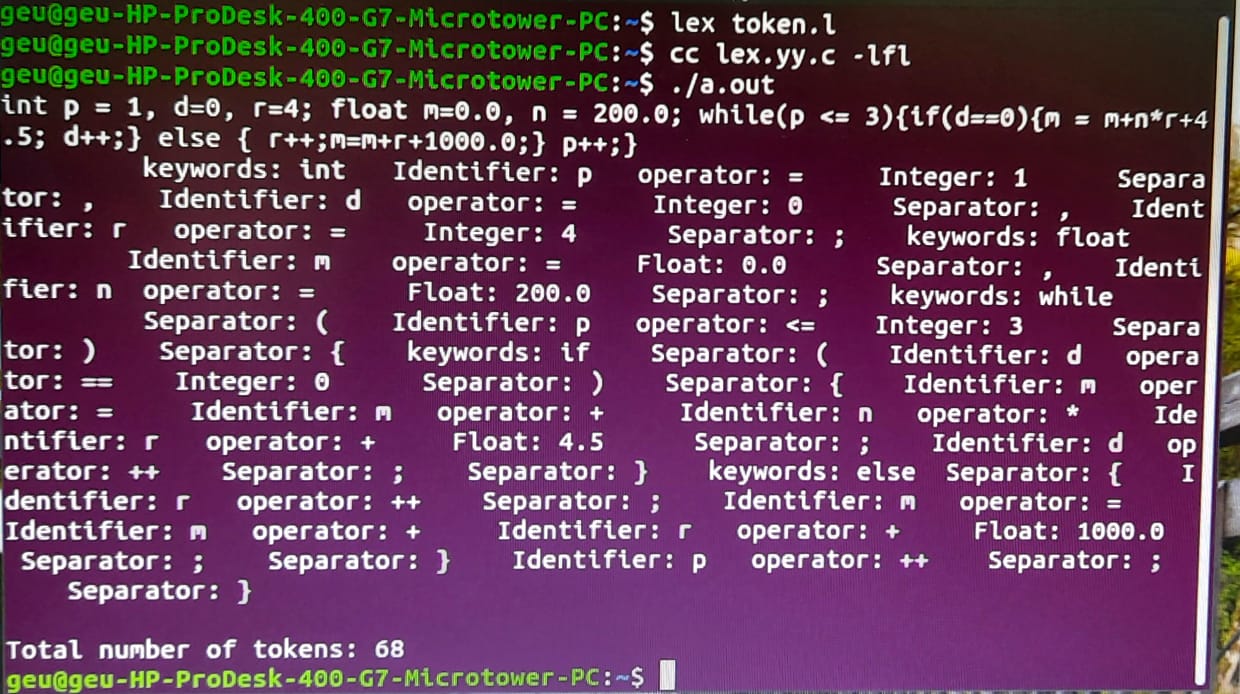
{

yylex();

printf("\nTotal number of tokens: %d\n",n);

return 0;

**OUTPUT:**

****

**PROGRAM-05**

**Design a LEX Code to count and print the number of total characters, words, white spaces in given ‘Input.txt’ file.**

**CODE:**

%{

int n,w,c;

%}

%%

\n {n++;}

[^ \n\t]+ {w++;c=c+yyleng;}

. c++;

%%

int main()

{

extern FILE \*yyin;

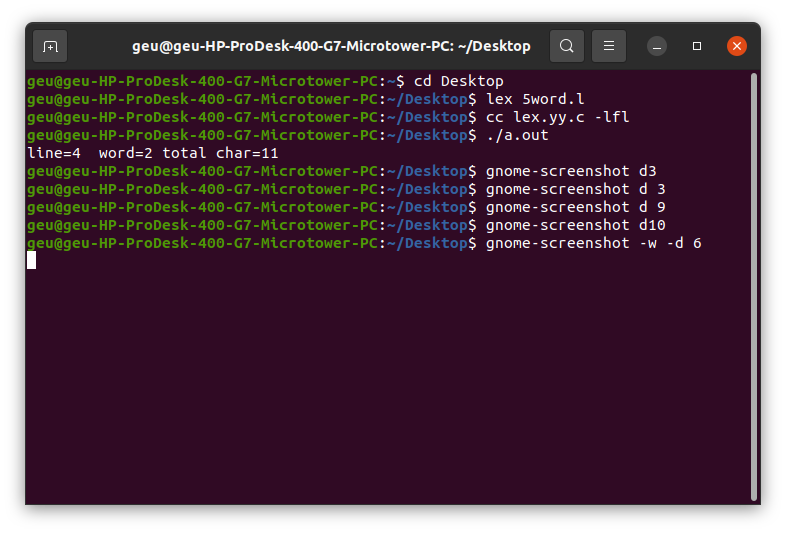
yyin=fopen("input.txt","r");

yylex();

printf("line=%d word=%d total char=%d \n",n,w,c);

}

**OUTPUT:**

****

**PROGRAM-06**

**Design a LEX Code to replace white spaces of ‘Input.txt’ file by a single blank character into ‘Output.txt’ file.**

**CODE:**

%{

%}

%%

[ \n\t]+ fprintf(yyout," ");

. fprintf(yyout,"%s",yytext);

%%

int main()

{

extern FILE \*yyin,\*yyout;

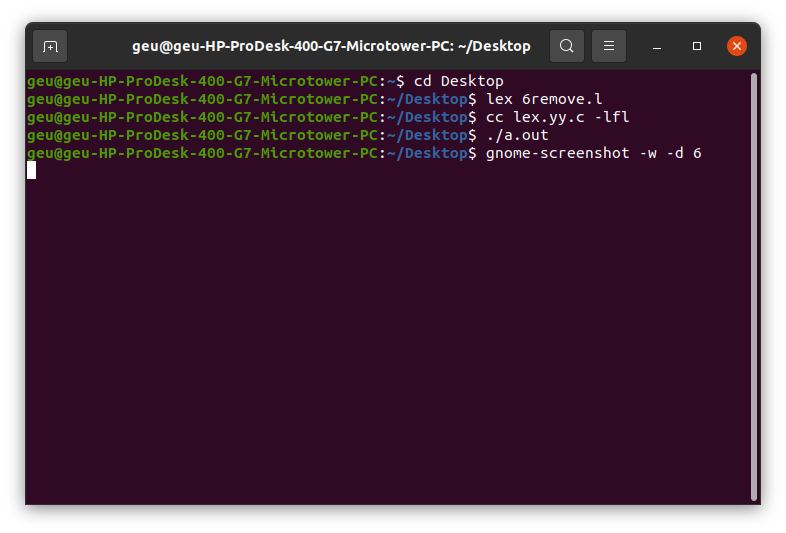
yyin=fopen("input.txt","r");

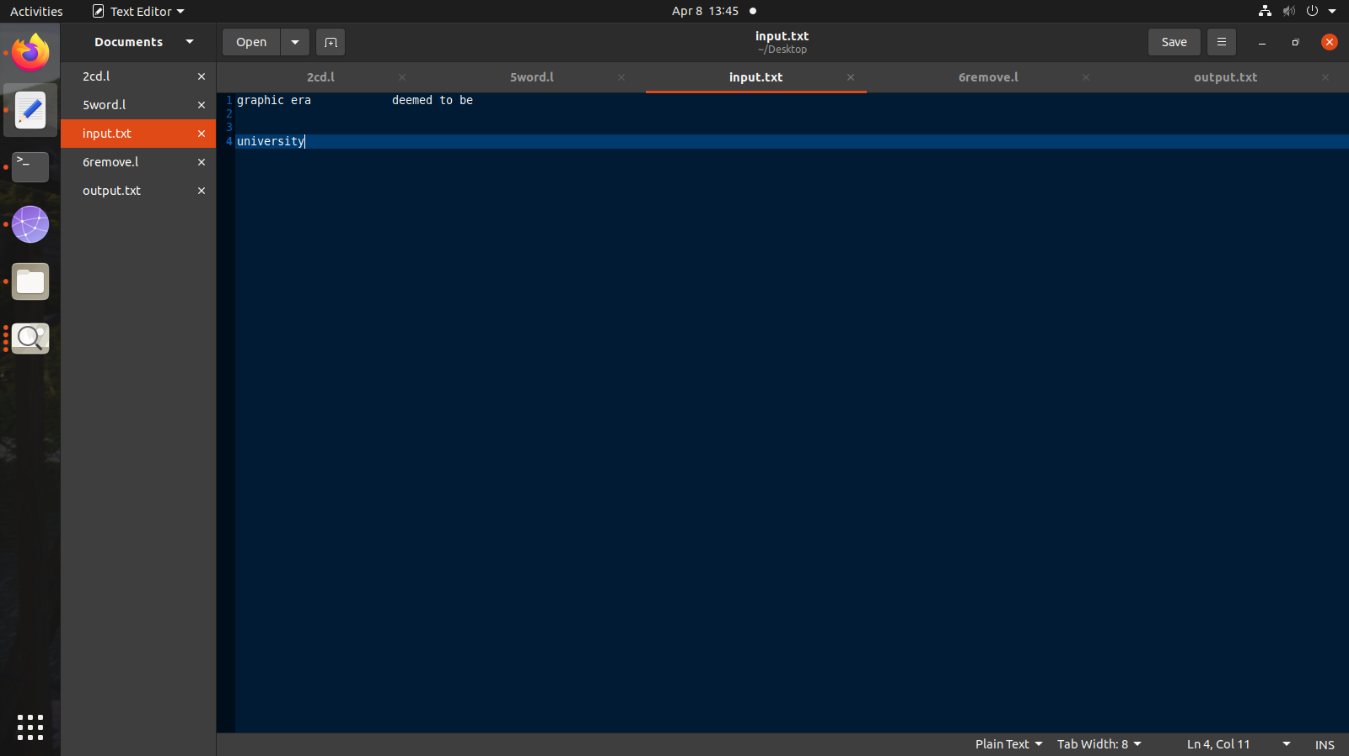
yyout=fopen("output.txt","w");

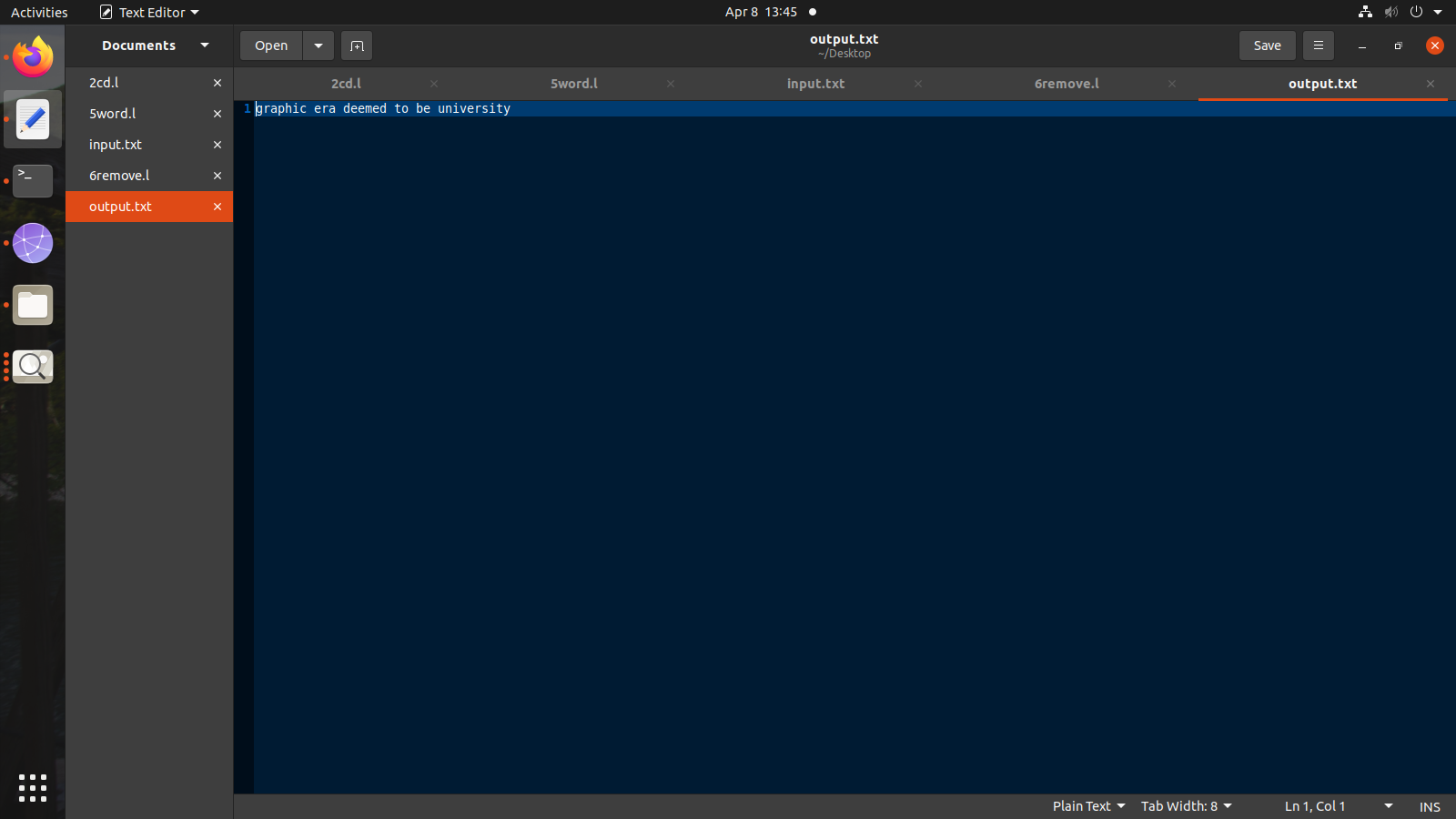
yylex();

}

**OUTPUT:**

****

****

****

**PROGRAM-07**

**Design a LEX Code to remove the comments from any C-Program given at run-time and store into ‘out.c’ file.**

**CODE:`**

%{

#include<stdio.h>

%}

%%

/Regular expression for single line comment/

\/\/(.\*) {};

/Regular expression for multi line comment/

\/\(.\n).\\*\/  {};

%%

/call the yywrap function/

int yywrap()

{

return 1;

}

int main ()

{

yyin=fopen("input6.c","r");

yyout=fopen("out.c","w");

yylex();

return 0;

}

**OUTPUT:**

****

**PROGRAM-08:**

**Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.**

**CODE:**

%{

#include<stdio.h>

%}

%%

\<[^>]\*\> fprintf(yyout,"%s\n",yytext);

.|\n;

%%

int yywrap()

{

return 1;

}

int main()

{

yyin=fopen("input7.html","r");

yyout=fopen("output7.txt","w");

yylex();

return 0;

}

**OUTPUT:**

