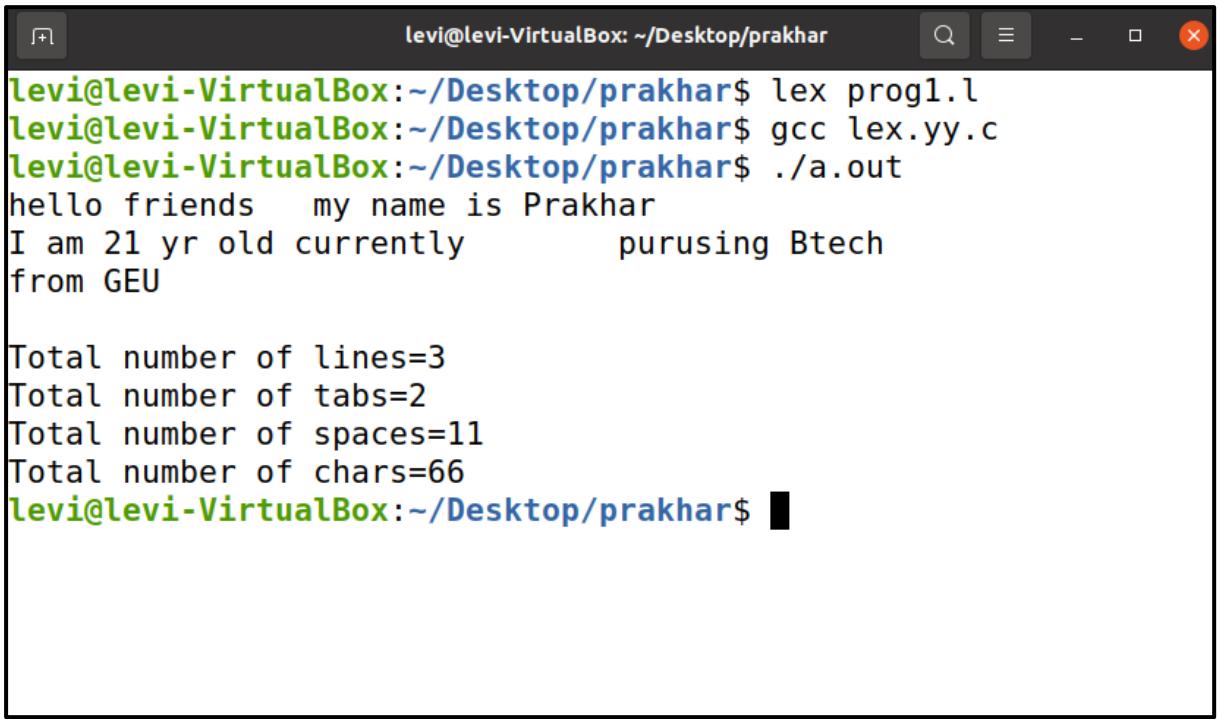
1. **Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.**

Code:-

```
%{
#include<stdio.h>
int n=0,m=0,t=0,c=0;
%}
%%
\n {n++;}
\t {m++;}
[ ] {t++;}
. {c++;}
%%
int  yywrap(){return 1;}
int main()
{
        yylex();
        printf("\nTotal number of lines=%d",n);
        printf("\nTotal number of tabs=%d ",m);
        printf("\nTotal number of spaces=%d",t);
        printf("\nTotal number of chars=%d\n",c);
        return 0;
}
```

Output:-



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

Code:-

```
%{
#include<stdio.h>
int c=0;
%}
%%
^[a-zA-Z_$][a-zA-Z0-9_$]* {c++; printf("\t Valid Identifier = %s\n",yytext);}
.* {printf("\t Invalid Identifier = %s\n",yytext);};
%%
int yywrap(){return 1;}
int main()
{
        yylex();
        printf("\nTotal number of valid Identifier = %d \n",c);
}
```

Output:-

```
levi@levi-VirtualBox: ~/Desktop/prakhar
levi@levi-VirtualBox:~/Desktop/prakhar$ lex prog2.l
levi@levi-VirtualBox:~/Desktop/prakhar$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/prakhar$ ./a.out
bunny
        Valid Identifier = bunny

honney bunny
        Invalid Identifier = honney bunny

honey_bunny
        Valid Identifier = honey_bunny

honey2bunny
        Valid Identifier = honey2bunny


Total number of valid Identifier = 3
levi@levi-VirtualBox:~/Desktop/prakhar$
```

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

Code:-

```
%{
#include<stdio.h>
int m=0,n=0;
%}
%%
[0-9]+ {m++; printf("\t Integer = %s\n",yytext);}
[0-9]*"."[0-9]+ {n++; printf("\t Float = %s\n",yytext);}
. ;
%%
int yywrap(){return 1;}
int main()
{
        yylex();
        printf("\nTotal number of Integer = %d \n",m);
        printf("\nTotal number of Float = %d \n",n);
}
```

Output:-

```
levi@levi-VirtualBox: ~/Desktop/prakhar
levi@levi-VirtualBox:~/Desktop/prakhar$ lex prog3.l
levi@levi-VirtualBox:~/Desktop/prakhar$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/prakhar$ ./a.out
69
        Integer = 69

6.9
        Float = 6.9

6969
        Integer = 6969


Total number of Integer = 2

Total number of Float = 1
levi@levi-VirtualBox:~/Desktop/prakhar$
```

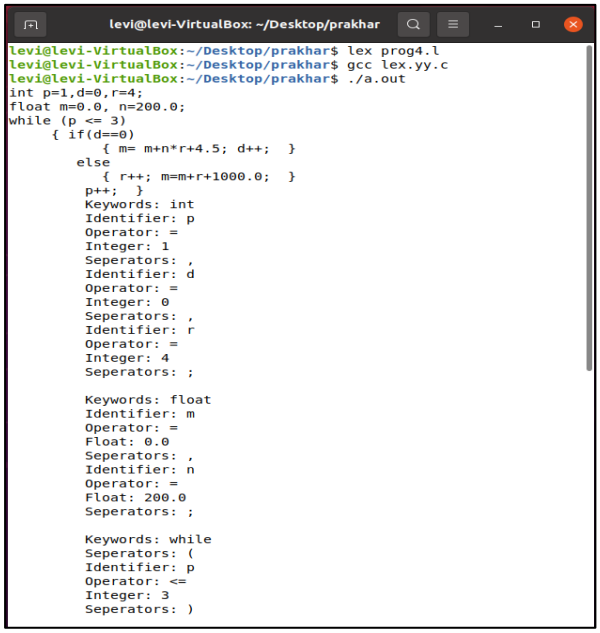
4. **Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS) the following C-fragment:**

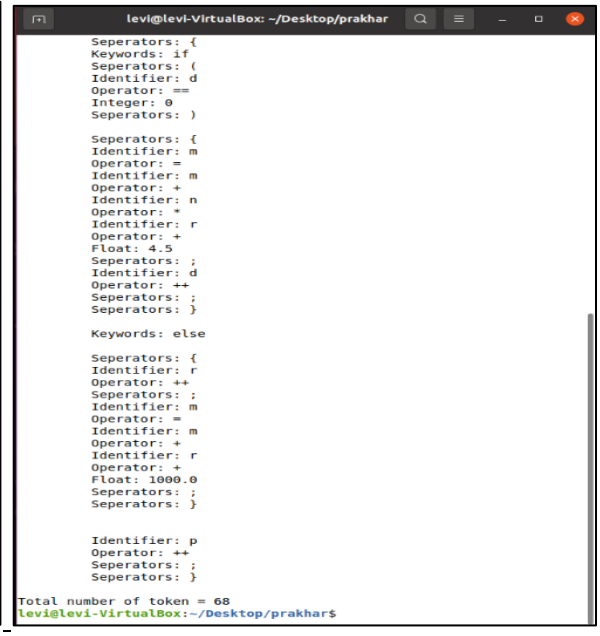
```
int p=1,d=0,r=4;
float m=0.0, n=200.0;
while (p <= 3)
{ if(d==0)
      { m= m+n*r+4.5; d++;  }
  else
     { r++; m=m+r+1000.0;  }
       p++;  }
```

Code:-

```
%{
#include<stdio.h>
int n=0;
%}
%%
"while"|"if"|"else"|"int"|"float" {n++; printf("\n\t Keywords: %s",yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {n++; printf("\n\t Identifier: %s",yytext);}
"<="|"=="|"="|"++"|"-"|"*"|"+" {n++; printf("\n\t Operator: %s",yytext);}
"("|")"|"{"|"}"|","|";" {n++; printf("\n\t Seperators: %s",yytext);}
[0-9]*"."[0-9]+ {n++; printf("\n\t Float: %s",yytext);}
[0-9]+ {n++; printf("\n\t Integer: %s",yytext);}
. ;
%%
int yywrap(){return 1;}
int main()
{
        yylex();
        printf("\nTotal number of token = %d \n",n);
}
```

Output:-



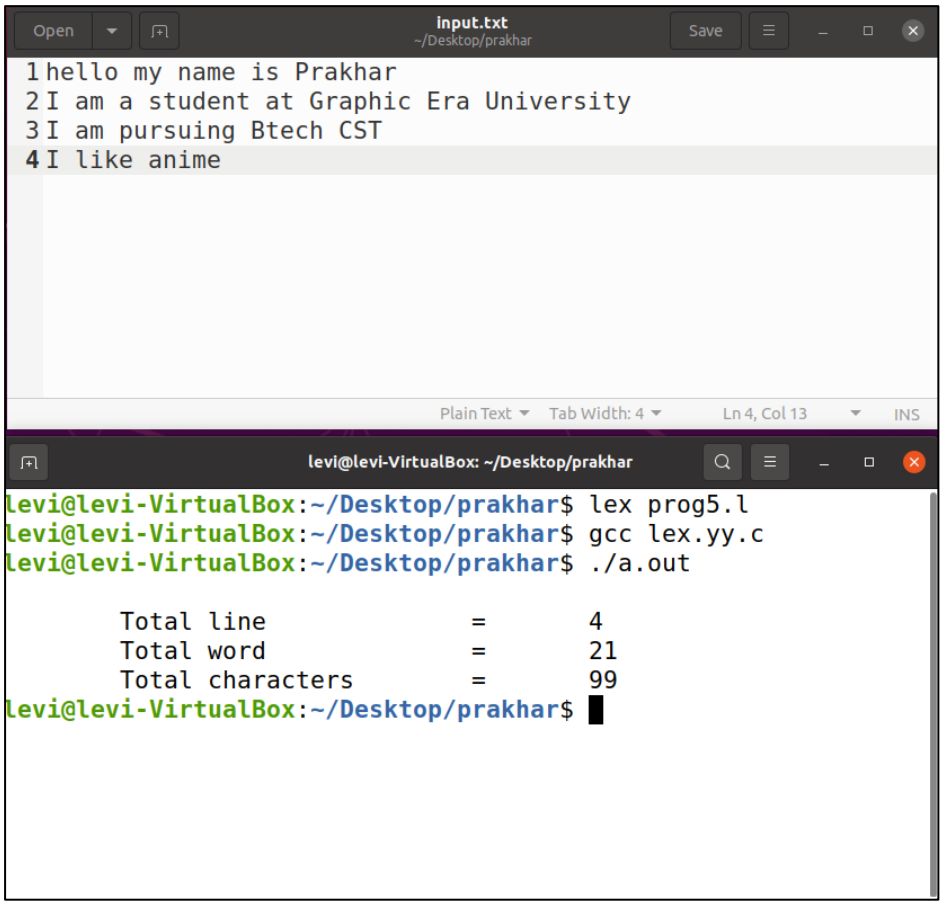


5. **Design a LEX Code to count and print the number of total characters, words, white spaces in given 'Input.txt' file.**

Code:-

```
%{
#include<stdio.h>
int n=0,w=0,c=0;
%}
%%
\n {n++;}
[^ \n\t]+ {w++;c=c+yyleng;}
. c++;
%%
int yywrap(){return 1;}
int main()
{
        extern FILE *yyin;
        yyin=fopen("input.txt","r");
        yylex();
        printf("\n\tTotal line\t\t=\t%d",n);
        printf("\n\tTotal word\t\t=\t%d",w);
        printf("\n\tTotal characters\t=\t%d\n",c);
}
```

Output:-

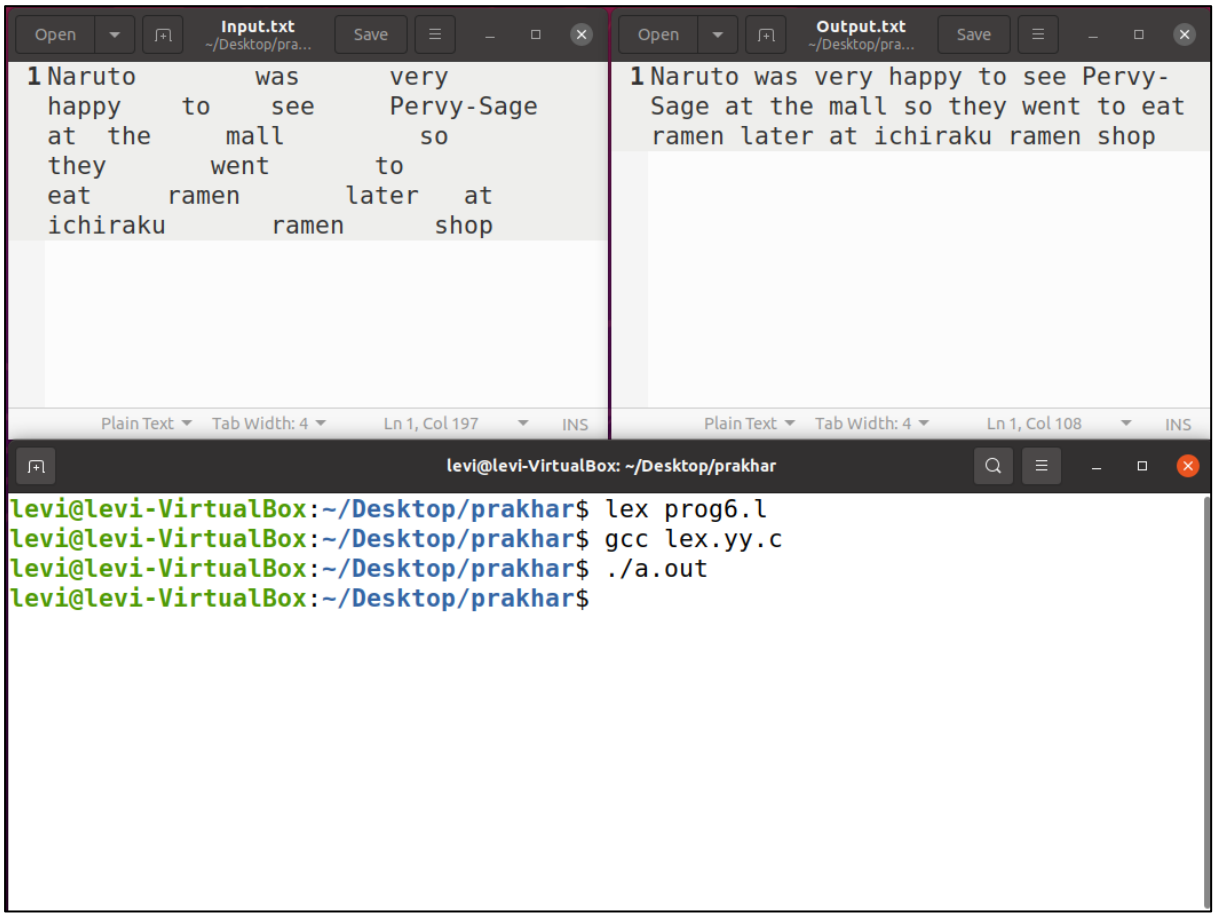


6. **Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file.**

Code:-

```
%{
#include<stdio.h>
%}
%%
[ \n\t]+ fprintf(yyout," ");
. fprintf(yyout,"%s",yytext);
%%
int yywrap(){return 1;}
int main()
{
        extern FILE *yyin,*yyout;
        yyin=fopen("Input.txt","r");
        yyout=fopen("Output.txt","w");
        yylex();
}
```

Output:-



7. **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>
%}
%%
\\(.*) {};
\\*(.*\n)*.*\*\/ {};
%%
int yywrap(){return 1;}
int main ()
{
        yyin=fopen("in.c","r");
        yyout=fopen("out.c","w");
        yylex();
}
```

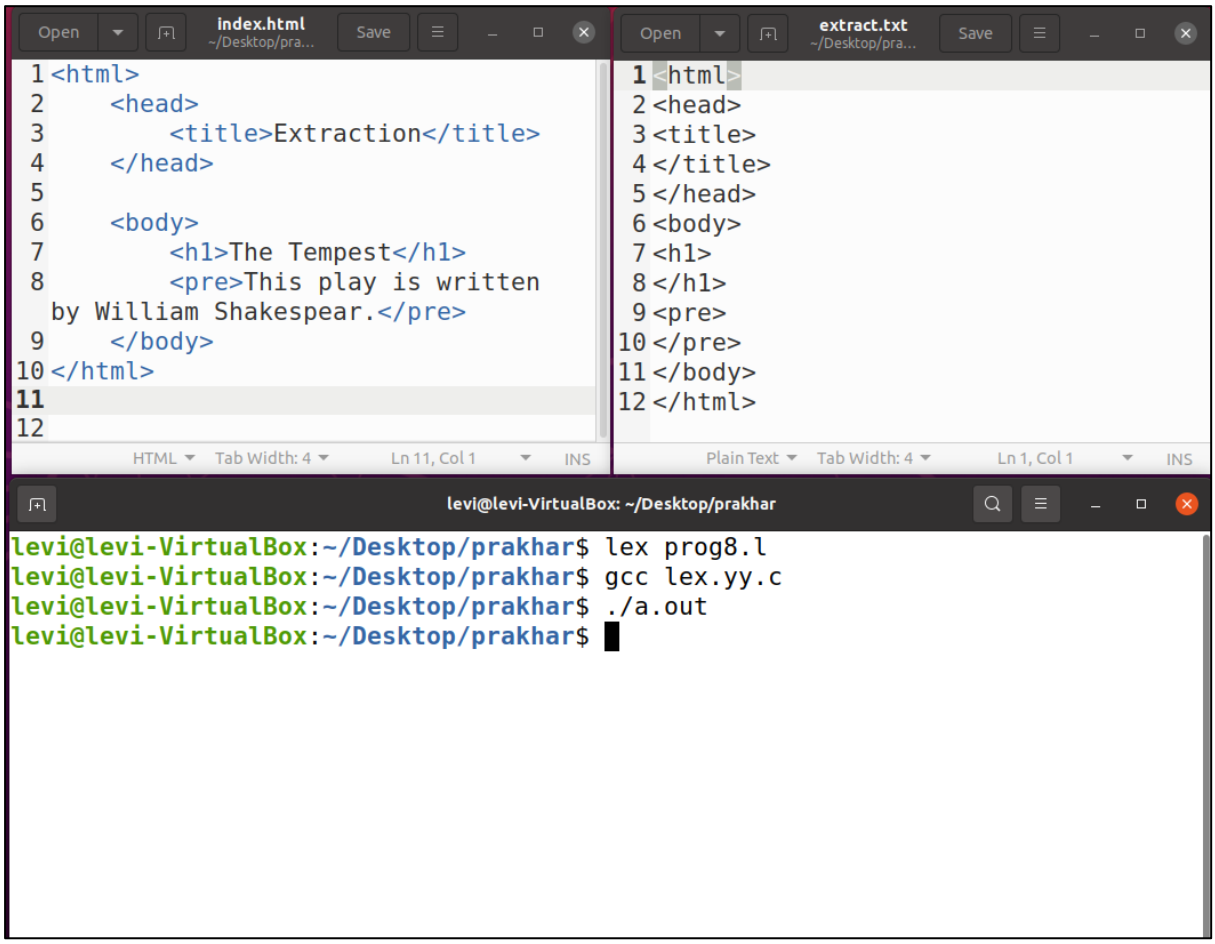
Output:-

8. **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("index.html","r");
        yyout=fopen("extract.txt","w");
        yylex();
        return 0;
}
```

Output:-



9. **Design a DFA in LEX Code which accepts string with even number of 0 over input alphabet {0,1}.**

Code:-

```
%{
#include<stdio.h>
#include<string.h>

%}
%s A end
%%

<INITIAL>0 BEGIN A;
<INITIAL>1 BEGIN INITIAL;
<INITIAL>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<A>0 BEGIN end;
<A>1 BEGIN A;
<A>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<end>1 BEGIN end;
<end>0 BEGIN A;
<end>\n BEGIN INITIAL; {printf("\naccepted\n");}
%%
int yywrap(void){return 0;}
int main()
{
yylex();
return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_even_0.l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
0101

accepted
00

accepted
01111111111101111110111111110

accepted
1111

not accepted
```

10. Design a DFA in LEX Code which accepts string starting with 0 and ending with 1 over input alphabet {0,1}.

Code:-

```
%{
#include<stdio.h>
#include<string.h>

%}
%s A end dead
%%

<INITIAL>0 BEGIN A;
<INITIAL>1 BEGIN dead;
<INITIAL>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<A>0 BEGIN A;
<A>1 BEGIN end;
<A>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<end>1 BEGIN end;
<end>0 BEGIN A;
<end>\n BEGIN INITIAL; {printf("\naccepted\n");}

<dead>0 BEGIN dead;
<dead>1 BEGIN INITIAL;
<dead>\n BEGIN INITIAL; {printf("\nnot accepted\n");}
%%
int yywrap(void){return 0;}
int main()
{
yylex();
return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_starting_0_end
ing_1.l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
0000

not accepted
1111

not accepted
0101

accepted
01110

not accepted
10000

not accepted
010101010111100001

accepted
```

11. Design a DFA in LEX Code which accepts string starting with 11 over input alphabet {0,1}.

Code:-

```
%{
#include<stdio.h>
#include<string.h>
%}
%s A end dead
%%

<INITIAL>0 BEGIN dead;
<INITIAL>1 BEGIN A;
<INITIAL>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<A>0 BEGIN dead;
<A>1 BEGIN end;
<A>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<end>1 BEGIN end;
<end>0 BEGIN end;
<end>\n BEGIN INITIAL; {printf("\naccepted\n");}

<dead>0 BEGIN dead;
<dead>1 BEGIN dead;
<dead>\n BEGIN INITIAL; {printf("\nnot accepted\n");}
%%
int yywrap(void){return 0;}
int main()
{
yylex();
return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_string_startin
g_11.l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
1010

not accepted
1100

accepted
001100

not accepted
01000001

not accepted
1111111111

accepted
```

12. Design a DFA in LEX Code which accepts string starting with odd 0 and even 1 over input alphabet {0,1}.

Code:-

```
%{
%}

%s A B C DEAD

%%
<INITIAL>1 BEGIN A;
<INITIAL>0 BEGIN B;
<INITIAL>[^01\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>1 BEGIN INITIAL;
<A>0 BEGIN C;
<A>[^01\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>1 BEGIN C;
<B>0 BEGIN INITIAL;
<B>[^01\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<C>1 BEGIN B;
<C>0 BEGIN A;
<C>[^01\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
int yywrap(){return 0;}
int main()
{
        yylex();
        return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_odd_0_even_1.l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
Enter String
10101
Not Accepted
00
Not Accepted
0
Accepted
1010101010101111101101
Not Accepted
0010101
Not Accepted
0010010010010
Accepted
```

13. Design a DFA in LEX Code which accepts string with even a and even b over input alphabet {a,b}.

Code:-

```
%{
#include<stdio.h>
#include<string.h>

%}
%s A B end
%%

<INITIAL>a BEGIN A;
<INITIAL>b BEGIN end;
<INITIAL>\n BEGIN INITIAL; {printf("\naccepted\n");}

<A>a BEGIN INITIAL;
<A>b BEGIN B;
<A>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<B>b BEGIN A;
<B>a BEGIN end;
<B>\n BEGIN INITIAL; {printf("\nnot accepted\n");}

<end>b BEGIN INITIAL;
<end>a BEGIN B;
<end>\n BEGIN INITIAL; {printf("\n not accepted\n");}
%%
int yywrap(void){return 0;}
int main()
{
yylex();
return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_even_a_even_b.
l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
abababababababababba

accepted
abbbbbbbb

not accepted
aaaa

accepted


accepted
abab

accepted
bbabb

not accepted
```

14. Design a DFA in LEX Code which accepts string with 3rd last word from RHS be a over input alphabet {a,b}.

Code:-

```
%{
%}
%s A B C D E F G DEAD
%%
<INITIAL>b BEGIN INITIAL;
<INITIAL>a BEGIN A;
<INITIAL>[^ab\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<A>b BEGIN F;
<A>a BEGIN B;
<A>[^ab\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<B>b BEGIN D;
<B>a BEGIN C;
<B>[^ab\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<C>b BEGIN D;
<C>a BEGIN C;
<C>[^ab\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Accepted\n");}
<D>b BEGIN G;
<D>a BEGIN E;
<D>[^ab\n] BEGIN DEAD;
<D>\n BEGIN INITIAL; {printf("Accepted\n");}
<E>b BEGIN F;
<E>a BEGIN B;
<E>[^ab\n] BEGIN DEAD;
<E>\n BEGIN INITIAL; {printf("Accepted\n");}
<F>b BEGIN G;
<F>a BEGIN E;
<F>[^ab\n] BEGIN DEAD;
<F>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<G>b BEGIN INITIAL;
<G>a BEGIN A;
<G>[^ab\n] BEGIN DEAD;
<G>\n BEGIN INITIAL; {printf("Accepted\n");}
<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
int yywrap(){ return 1;}
int main()
{
 yylex();
 return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_last_3rd_is_a.
l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
baab
Accepted
ab
Not Accepted
baaa
Accepted
```

15. Design a DFA in LEX Code to Identify and print Integer & Float Constants and Identifier.

Code:-

```
%{
%}
%s A B C DEAD
%%
<INITIAL>[0-9]+ BEGIN A;
<INITIAL>[0-9]+[.][0-9]+ BEGIN B;
<INITIAL>[A-Za-z_][A-Za-z0-9_]* BEGIN C;
<INITIAL>[^\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>[^\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Integer\n");}

<B>[^\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Float\n");}

<C>[^\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Identifier\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%

int yywrap(){return 1;}
int main()
{
yylex();
return 0;
}
```

Output:-

```
levi@levi-VirtualBox:~/Desktop/compiler-design$ lex dfa_integer_float_
identifier.l
levi@levi-VirtualBox:~/Desktop/compiler-design$ gcc lex.yy.c
levi@levi-VirtualBox:~/Desktop/compiler-design$ ./a.out
99
Integer
34
Integer
34.58
Float
happy
Identifier
happy_2
Identifier
2happy
Invalid
levi@levi-VirtualBox:~/Desktop/compiler-design$
```