

1. Write a lex program to recognize single line comment and multiple line comment

```
%{
#include<stdio.h>
int scount, mcount;
}%

%%
[/]{1}[/]{1}[a-zA-Z0-9]* { scount++;}
[/]{1}[*]{1}[a-zA-Z0-9]*[*]{1}[/]{1} {mcount++;}

%%

int main()
{
    FILE *fp;
    fp=fopen("test.c","r");
    if(fp==NULL)
    {
        printf("File cant open");
        return 0;
    }
    yyin=fp;
    yylex();
    printf("single line comments is=%d",scount);
    printf("\nMultiple line commentis=%d",mcount);
}
```

2. Write a lex program to recognize Valid arithmetic expression

Lab Program 1a

```
%{
#include<stdio.h>
int v=0,op=0,id=0;
}%

%%
[0-9][0-9]* {id++; printf("\nIdentifier:"); ECHO;}
[+|-|\*|/=] {op++; printf("\nOperartor:"); ECHO;}
"(" {v++;}
")" {v--;}
.|\\n {return 0;}

```

%%

```
int main()
{
    printf("Enter the expression:\n");
    yylex();
    if((op+1) ==id && v==0 )
    {
        printf("\n\nIdentifiers are:%d\nOperators are:%d\n",id,op);
        printf("\nExpression is Valid\n");
    }
    else
        printf("\nExpression is Invalid\n");
    return 1;
}
int yywrap()
{
    return 1;
}
```

3. Write a Yacc program to recognize Valid arithmetic expression

// Lab Program 1b

Yacc part:

```
%{
#include<stdio.h>
#include<stdlib.h>
%}

%token num
%left '+' '-'
%left '*' '/'

%%
input:exp {printf("%d\n",$$);exit(0);}
exp: exp '+' exp {$$=$1+$3;}
    |exp '-' exp {$$=$1-$3;}
    |exp '*' exp {$$=$1*$3;}
    |exp '/' exp { if($3==0){printf("Divide by Zero. Invalid
expression.\n");exit(0);}
```

```

        else $$=$1/$3;}
        |('exp'){$$=$2;}
        |num{$$=$1;};
%%

int yyerror()
{
    printf("Error. Invalid Expression.\n");
    exit(0);
}
int main()
{
    printf("Enter an expression:\n");
    yyparse();
}

```

Lex part:

```

%{
#include "y.tab.h"
extern yylval;
%}

%%
[0-9]+      {yylval=atoi(yytext);return num;}
[+\-\\*\V]  {return yytext[0];}
[]          {return yytext[0];}
[(]         {return yytext[0];}
.           {;}
\n          {return 0;}
%%

```

4. write a lex program to recognize whether a given sentence is simple or compound

```

%{
#include<stdio.h>
int flag=0;
%}

```

```
%%  
and |  
or |  
but |  
because |  
if |  
then |  
nevertheless  
{flag=1;}  
.  
\n { return 0; }  
%%
```

```
int main()  
{  
    printf("Enter the  
sentence:\n");  
    yylex();  
    if(flag==0)
```

```
printf("Simple  
sentence\n");  
    else
```

```
printf("compound  
sentence\n");  
}
```

```
int yywrap( )  
{  
    return 1;  
}
```

5. Write a Lex program to count the number of vowels and consonants in a given string.

```
%{
    #include<stdio.h>
    int vowels=0;
    int consonants=0;
}%
%%

[aeiouAEIOU] {vowels++;}
[a-zA-Z] {consonants++;}
%%

int yywrap()
{
    return 1;
}

main()
{
    printf("Enter the string ");
    yylex();
    printf("No. of vowels=%d\n No. of
consonants=%d\n",vowels, consonants);
}
```

6. Write a Lex program to count the number of 'printf' and 'scanf' statements in a c code. Replace them with 'writef' and 'readf' respectively. (consider command line arguments)



7. Write a Yacc program to accept the strings of the form $a^n b^n$ ($n > 0$)

Lex part:

```
%{  
    /* Definition section */  
    #include "y.tab.h"  
}%
```

```
/* Rule Section */  
  
%%  
  
[aA] {return A;}  
[bB] {return B;}  
\n {return NL;}  
. {return yytext[0];}  
%%
```

```
int yywrap()  
{  
    return 1;  
}
```

Yacc part:

```
%{  
    /* Definition section */  
    #include<stdio.h>  
    #include<stdlib.h>  
}%
```

```
%token A B NL  
  
/* Rule Section */  
  
%%
```

```
stmt: S NL { printf("valid string\n");  
        exit(0); }
```

```
;
```

```
S: A S B |
```

```
;
```

```
%%
```

```
int yyerror(char *msg)
```

```
{
```

```
    printf("invalid string\n");
```

```
    exit(0);
```

```
}
```

```
//driver code
```

```
main()
```

```
{
```

```
    printf("enter the string\n");
```

```
    yyparse();
```

```
}
```

Write a Yacc program to accept the strings of the form $a^n b$ (input n value)

//Check Lab Prog -2

2.1

```
%{
```

```
    #include<stdio.h>
```

```
    #include "y.tab.h"
```

```
%}
```

```
%%
```



```

a {return A;}

b {return B;}

[\n] return '\n';

%%

```

2.y

```

%{
    #include<stdio.h>
    #include<stdlib.h>
    #include<string.h>
%}

%token A B

%%

input: S {printf("Parsed Successfully\n");exit(0);}

S:A S1 B | B

S1:A S1

|;

%%

extern FILE *yyin;

int main()
{
    char str[30];
    int n=0;
    for(int i=0;i<30;i++)
        str[i]='\0';
    printf("Enter n:"); scanf("%d",&n);
    for(int i=0;i<n;i++)
        strcat(str,"a");

```

```

    strcat(str,"b");
    strcat(str,"\0");
    printf("%s",str); printf("\n");
    FILE *input = fopen("input.txt","w");
    fprintf(input,"%s", str);
    fclose(input);
    input=fopen("input.txt","r");
    yyin=input;
    yyparse();
}

int yyerror()
{
    printf("Error");exit(0);
}

int yywrap()
{
    return 1;
}

```

8. Write a Lex program to check whether the number is odd or even

```

%{
#include<stdio.h>
int i;
%}

%%

[0-9]+      {i=atoi(yytext);
             if(i%2==0)
                 printf("Even");
             else
                 printf("Odd");}

%%

```

```

int yywrap() {}

/* Driver code */
int main()
{

    yylex();
    return 0;
}

```

9. Lex program to take input from a file, remove multiple spaces, newline and tab & write output in a separate file

```

%{

    /*Definition section */

    %

}

/* Rule: whenever space, tab or
newline is encountered, remove it*/

%%

[ \n\t]+ {fprintf(yyout, "%s");}
.      { fprintf(yyout, "%s", yytext); }

%%

// driver code

int main()
{

    /* yyin and yyout as pointer
of File type */

    extern FILE *yyin, *yyout;

    /* yyin points to the file input.txt
and opens it in read mode*/

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

    /* yyout points to the file output.txt
and opens it in write mode*/

```

```

yyout = fopen("Output.txt", "w");
yylex();
return 0;
}

```

10. Lex program to copy the content of one file to another file

```

%{
#include<stdio.h>
#include<string.h>
char line[100];
%}

/* Rule Section */

/* Rule 1 writes the string stored in line
   character array to file output.txt */
/* Rule 2 copies the matched token
   i.e every character except newline character
   to line character array */

%%

['\n'] { fprintf(yyout,"%s\n",line);}
(.*?) { strcpy(line,yytext);}
<<EOF>> { fprintf(yyout,"%s",line); return 0;}

%%

int yywrap()
{
    return 1;
}

/* code section */

int main()

```

```

{
    extern FILE *yyin, *yyout;

    /* open the source file
       in read mode */
    yyin=fopen("input.txt","r");

    /* open the output file
       in write mode */
    yyout=fopen("output.txt","w");
    yylex();
}

```

11. Lex program to add line numbers to a given file

```

%{
#include<stdio.h>

int line_number = 1; // initializing line number to 1
%}

/* simple name definitions to simplify
the scanner specification name
definition of line*/
line  .*\\n
%%

{line}  { printf("%10d %s", line_number++, yytext); }

/* whenever a line is encountered increment count*/
/* 10 specifies the padding from left side to
       present the line numbers*/
/* yytext The text of the matched pattern is stored

```

in this variable (char*)*/

%%

int yywrap()

{

return 1;

}

int main(int argc, char*argv[])

{

extern FILE *yyin; // yyin as pointer of File type

yyin = fopen("testtest.c", "r"); /* yyin points to the file

testtest.c and opens it

in read mode.*/

yylex();

return 0;

}