Final Practical File

Name: Ravish Ranjan

Paper: System Programming

Roll no.: 21058570040

Sem: semester 5th

Que1 Write a Lex program to count the number of lines and characters in the input file.

Ans

```
%option noyywrap

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

%%
\n {linecount++;}
[a-zA-Z0-9 ]* {charcount+=yyleng;}

%%
int main(){
    yyin = fopen("text.txt","r");
    yylex();
    printf("charcount : %d\n",charcount);
    printf("linecount : %d",linecount);
    return 0;
}
```

#text.txt file

```
ys prog > text.txt
1   hello
2   how are you
3   i am under the water
4   please help me
5   lulululululu
6   call 911
```

#Output

```
C:\Users\COMP46\Desktop\4076\sys prog>flex pro1.l

C:\Users\COMP46\Desktop\4076\sys prog>gcc lex.yy.c

C:\Users\COMP46\Desktop\4076\sys prog>a.exe
charcount : 70
linecount : 5
C:\Users\COMP46\Desktop\4076\sys prog>
```

Que 2 Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e, and so on z by c.

Ans

```
%option noyywrap
%{
#include <stdio.h>
%}

%%
[a-zA-Z]* {
    char string[100] = "";
    for (int i = 0;i < yyleng;i++){
        int y = yytext[i];
        if ((y>=120)&&(y<=122)){
            y -= 26;
        }
        else if ((y>=88)&&(y<=90)){
            y -= 26;
        }
        string[i] = (char) (y+3);
    }
    printf("\nYour sting ( %s ) is now : %s",yytext,string);
}

%%
int main(){
    yylex();
    return 0;
}
```

#Output

```
C:\sem5\vs code\sysprog>flex pro2.l

C:\sem5\vs code\sysprog>gcc lex.yy.c

C:\sem5\vs code\sysprog>a.exe
This is a sample text

Your sting ( This ) is now : Wklv
Your sting ( is ) is now : lv
Your sting ( a ) is now : d
Your sting ( sample ) is now : vdpsoh
Your sting ( text ) is now : whaw

C:\sem5\vs code\sysprog>
```

Que3 Write a Lex program that finds the longest word (defined as a contiguous string of upper-and lower-case letters) in the input.

Ans

```
%option noyywrap

%{
#include <stdio.h>
int longest = 0;
char string[100] = "";
%}

%%
[a-zA-Z0-9 ]* {
    if (yyleng > longest){
        longest = yyleng;
        for (int i = 0; i< yyleng;i++){
            string[i] = yytext[i];
        }
        string[yyleng] = '\0';
    }
}

%%
int main(){
    yyin = fopen("text.txt","r");
    yylex();
    printf("Longest word : %s\nIt's size : %d",string,longest);
    return 0;
}
```

#text.txt file

```
Lorem
ipsum
dolor
sit
amet
consectetur
adipisicing
elit
Quos
odit
eum
voluptates
eaque
maiores
eligendi
magni
Itaque
eaque
praesentium
quisquam
```

#Output

```
C:\Users\COMP46\Desktop\4076\sys prog>flex pro3.l

C:\Users\COMP46\Desktop\4076\sys prog>gcc lex.yy.c

C:\Users\COMP46\Desktop\4076\sys prog>a.exe



Longest word : consectetur
It's size : 11
C:\Users\COMP46\Desktop\4076\sys prog>
```

Que 4 Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.

Ans

```
%option noyywrap
%{
#include <stdio.h>
char keywd,integ,flts,inden,opers,comms;
%}

integ (\+|\-)?[0-9]+
flts {integ}\.[0-9]+
opers [\<\>\.\*\+\/&%=\!]{1,2}
inden [a-zA-Z_][a-zA-Z0-9]*
keywd
(if|else|const|double|int|float|short|struct|unsigned|break|continue|for|long|sig
ned|switch|void|case|char|do|extern|return|static|union|while)
comms ^(\/\/|\/\*).*

%%
```

```
{keywd} {printf(">> a keyword\n");}
{integ} {printf(">> an integer\n");}
{flts} {printf(">> a float\n");}
{opers} {printf(">> an operator\n");}
{inden} {printf(">> an identifier\n");}
{comms} {printf(">> a comment\n");}

%%
int main(){
    yylex();
    return 0;
}
```

#Output

```
C:\sem5\vs code\sysprog>flex pro2.l

C:\sem5\vs code\sysprog>gcc lex.yy.c

C:\sem5\vs code\sysprog>a.exe
case
>> a keyword

123
>> an integer

123.345
>> a float

!
>> an operator

asdhb234
>> an identifier

//ashjb
>> a comment

/* jnasd*/
>> a comment

_akjhbdka
>> an identifier

+123
>> an integer

-1238
>> an integer
```

Que5 Write a Lex program to count the number of identifiers in a C file.

Ans

```
%option noyywrap
%{
#include <stdio.h>
char identifier;
int iden_count = 0;
%}

identifier ^[a-zA-Z_][a-zA-Z0-9_]*

%%
{identifier} {
    iden_count++ ;
    printf(yytext);
}

%%
int main(){
    yyin = fopen("inp.txt","r");
    yylex();
    printf("\nTotal no. of identifiers : %d",iden_count);
    return 0;
}
```

#input file

```
Lorem
123hello
String
buffer
_gotit
Page
0pages
_6_
__
```

#Output

```
C:\sem5\vs code\sysprog>flex pro5.l

C:\sem5\vs code\sysprog>gcc lex.yy.c

C:\sem5\vs code\sysprog>a.exe
Lorem
123hello
String
buffer
_gotit
Page
0pages
_6_
__
Total no. of identifiers : 7
C:\sem5\vs code\sysprog>
```

Que6 Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

Ans

```
%option noyywrap
%{
#include <stdio.h>
int word_count,char_count,space_count,line_count = 0;
%}

%%
[ \t] {
    space_count++;
    char_count+=yyleng;
}
[\n] {
    line_count++;
    char_count+=yyleng;
}
[^\t\n ]+ {
    word_count++;
    char_count+=yyleng;
}

%%
int main(){
    yyin = fopen("inp.txt","r");
    yylex();
    printf("\nNo. of words : %d",word_count);
    printf("\nNo. of char  : %d",char_count);
    printf("\nNo. of space : %d",space_count);
    printf("\nNo. of line  : %d",line_count);
    return 0;
}
```

#input file

sysprog > inp.txt

```
Lorem ipsum dolor sit amet consectetur adipisicing elit.
Unde eius, magnam repellat veniam vero ratione suscipit in tempore consequuntur praesentium.
```

Que7 Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output.

abcd
abc
ab
a

Ans

```
%option noyywrap
%{
#include <stdio.h>
%}

%%
[a-zA-Z]{4,4} {
    char* string = yytext;
    printf("\n");
    for (int i = 4;i>0;i--){ // 4 3 2 1
        for (int j = 0; j < i;j++){
            printf("%c",string[j]);
        }
        printf("\n");
    }
}

%%
int main(){
    yylex();
    return 0;
}
```

#Output

```
C:\Users\COMP46\Desktop\4076\sys prog>flex pro7.l

C:\Users\COMP46\Desktop\4076\sys prog>gcc lex.yy.c

C:\Users\COMP46\Desktop\4076\sys prog>a.exe
back

back
bac
ba
b

GOAT

GOAT
GOA
GO
G


C:\Users\COMP46\Desktop\4076\sys prog>
```

Que8 A program in Lex to recognize a valid arithmetic expression.

Ans

//yc8.l

```
%option noyywrap
%{
    #include<stdio.h>
    #include<stdlib.h>
    int c,d,bo=0,bc=0;
%}

operand [a-zA-Z0-9]+
operator [+\-\/*]

%%
{operator} {
    d++;
    printf("%s is an operator \n",yytext);
    }

{operand} {
    c++;
    printf("%s is an operand \n",yytext);
}
```

```
"(" {
    if(bc<=bo){
        bo++;
    }
}

")" {
    bc++;
}

\n {
    if(bo==bc&&c>d){
        printf("valid exp");
    } else{
        printf("invalid exp");
    }
    exit(0);
}

%%
void main(){
    yylex();
}
```

//Output

```
C:\sem5\vs code\sp>a
(a+b
a is an operand
+ is an operator
b is an operand
invalid exp
C:\sem5\vs code\sp>a
(a+b)
a is an operand
+ is an operator
b is an operand
valid exp
C:\sem5\vs code\sp>
```

Que9 Write a YACC program to find the validity of a given expression (for operators + - * and /)

Ans

//yc9.y

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

%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%
stmt:e {printf("\nresult : %d",$$);}
e: e '+' e {$$ = $1 + $3;}
    | e '-' e {$$ = $1 - $3;}
    | e '*' e {$$ = $1 * $3;}
    | e '/' e {$$ = $1 / $3;}
    | e '%' e {$$ = $1 % $3;}
    | '(' e ')' {$$ = $2;}
    | NUMBER ;

%%
int main(){
    printf("Enter expression (+ / * - %) : ");
    yyparse();
    printf("\nValid expression :)");
    return 0;
}
int yyerror(){
    printf("\nInvalid expression :(");
    return 0;
}
```

//yc9.l

```
%{
    #include <stdio.h>
    #include "yc1.tab.h"
    extern int yylval;
%}
```

```
%%
[0-9]+ {
    yylval = strtol(yytext,NULL,0);
    return NUMBER;
}
\n {return 0;}
. {return yytext[0];}

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

//Output

```
C:\sem5\vs code\sp>yc1.exe
Enter expression (+ / * - ) : 1+5

result : 6
Valid expression :)
C:\sem5\vs code\sp>yc1.exe
Enter expression (+ / * - ) : 1

result : 1
Valid expression :)
C:\sem5\vs code\sp>yc1.exe
Enter expression (+ / * - ) : s

Invalid expression :(
Valid expression :)
C:\sem5\vs code\sp>
```

Que10 A Program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.

Ans

//yc10.l

```
%{
    #include <stdio.h>
    #include "yc10.tab.h"
%}

%%
[a-z]+[0-9]+ {return VALID;}
. {return INVALID;}
```

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

// yc10.y

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

%token VALID
%token INVALID

%%
valid :v {printf(" : Valid variable\n")}
v:VALID;
invalid :i {printf(" : Invalid variable\n")}
i:INVALID;

%%
int main(){
    printf("Enter variable : ");
    yyparse();
    return 0;
}
int yyerror(){
    printf("\nerror");
    return 0;
    exit(1);
}
```

//Output

```
PS C:\sem5\vs code\sp> ./yc10
Enter variable : qwe10
 : Valid variable

PS C:\sem5\vs code\sp>
```

Que11 A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

Ans

//yc11.l

```
%{
    #include <stdio.h>
    #include "yc1.tab.h"
    extern int yylval;
%}

%%
[0-9]+ {
    yylval = strtol(yytext,NULL,0);
    return NUMBER;
}
\n {return 0;}
. {return yytext[0];}

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

// yc11.y

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

%token NUMBER
%left '+' '-'
%left '*' '/'
%left '(' ')'

%%
stmt:e {printf("\nresult : %d",$$);}
e: e '+' e {$$ = $1 + $3;}
    | e '-' e {$$ = $1 - $3;}
    | e '*' e {$$ = $1 * $3;}
    | e '/' e {$$ = $1 / $3;}
    | '(' e ')' {$$ = $2;}
    | NUMBER ;

%%
int main(){
    printf("Enter expression (+ / * - %) : ");
    yyparse();
    printf("\nValid expression :)");
```

```
    return 0;
}
int yyerror(){
    printf("\nInvalid expression :(");
    return 0;
}
```

//Output

```
PS C:\sem5\vs code\sp> ./yc1
Enter expression (+ / * - ) : 45-85

result : -40
Valid expression :)
PS C:\sem5\vs code\sp> ./yc1
Enter expression (+ / * - ) : 45/9

result : 5
Valid expression :)
PS C:\sem5\vs code\sp> ./yc1
Enter expression (+ / * - ) : 16+9

result : 25
Valid expression :)
PS C:\sem5\vs code\sp> ./yc1
Enter expression (+ / * - ) : 9*5

result : 45
Valid expression :)
PS C:\sem5\vs code\sp>
```

Que12 Program in YACC to recognize the strings “ab”, “aabb”, ”aaabbb”,… of the language ($an\ bn$, n>=1).

Ans

//yc12.l

```
%{
    #include "yc12.tab.h"
%}

%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
. {return yytext[0];}

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

```
}
```

// yc12.y

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

%token A B NL

%%
stmt: S NL { printf("valid string\n"); exit(0); } ;
S: A S B | ;

%%
int yyerror(char *msg) {
    printf("invalid string\n");
    exit(0);
}
main() {
    printf("enter the string\n");
    yyparse();
}
```

//Output

```
PS C:\sem5\vs code\sp> ./yc12
enter the string
aaab
invalid string
PS C:\sem5\vs code\sp> ./yc12
enter the string
abbb
invalid string
PS C:\sem5\vs code\sp> ./yc12
enter the string
aaabbb
valid string
PS C:\sem5\vs code\sp> ./yc12
enter the string
abab
invalid string
PS C:\sem5\vs code\sp>
```

Que13 Program in YACC to recognize the language (anb , n>=10). (Output to say input is valid or not)

Ans

//yc13.l

```
%{
    #include "yc13.tab.h"
    extern int yylval;
%}

A [a]{10,}
B [b]

%%
{A} {yylval=yytext[0];return A;}
{B} {yylval=yytext[1];return B;}
\n {return 0;}
. {return yytext[0];}

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

// yc13.y

```
%{
    #include<stdio.h>
    #include<stdlib.h>
    int yylex(void);
%}

%token A B

%%
expr: s B {printf("\n valid string");};
s : s A | A ;

%%
int main(){
    printf(" Enter the string \n");
    yyparse();
    return 0;
}
int yyerror(){
    printf(" Invalid: Not a part of the language - a^n b \n");
}
```

//Output

```
PS C:\sem5\vs code\sp> ./yc13
 Enter the string
aaaab
 Invalid: Not a part of the language - a^n b
PS C:\sem5\vs code\sp> ./yc13
 Enter the string
aaaaaaaaab
 Invalid: Not a part of the language - a^n b
PS C:\sem5\vs code\sp> ./yc13
 Enter the string
aaaaaaaaaab

 valid string
PS C:\sem5\vs code\sp> ./yc13
 Enter the string
abababa
 Invalid: Not a part of the language - a^n b
PS C:\sem5\vs code\sp>
```