Activity Based Assessment

System Software and Compiler Design (21CS63)

Semester / Sec:	VI 'A'	**Branch:**	CSE
Course Instructor:	Dr. Chethana H T	**Academic: Year**	2023-24

Team Members

Sl No.	USN	NAME
1	4VV21CS057	DIVYASHREE B
2	4VV21CS058	HANSIKA M
3	4VV21CS060	HARSHIL M
4.	4VV21CS062	HEMANTH KUMAR S P

Activity Description: Students shall design and develop solutions using Lex and Yaac in a team of maximum 4 members. Each team shall come up with solutions for the below problem statements.

1) Write a LEX program to recognize valid arithmetic expression. Identifiers in the expression could be only integers and operators could be + and *. Count the identifiers & operators present and print them separately.
2) Write YACC program to evaluate arithmetic expression involving operators: +, -, *, and /.
3) Develop, Implement and execute a program using YACC tool to recognize all strings ending with b preceded by n a's using the grammar a n b (note: input n value).
4) Write a LEX program to eliminate comment lines in a C program and copy the resulting program into a separate file.
5) Write YACC program to recognize valid identifier, operators and keywords in the given text (C program) file.
6)Write LEX program to count the number of characters, words, spaces and lines in each input file.
7) Write a LEX program to recognize whether a given sentence is simple or compound.
8) Write a YACC program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.

Problem Statement:

1) Write a LEX program to recognize valid arithmetic expression. Identifiers in the expression could be only integers and operators could be + and *. Count the identifiers & operators present and print them separately.

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

int identifier_count = 0;
int operator_count = 0;
int is_valid = 1;

%}

/* Regular expressions for identifiers and operators */
integer     [0-9]+
operator    [+\*]

%%

{integer}   { identifier_count++; }
{operator}  { operator_count++; }
[\ \t\n]    { /* Ignore whitespace */ }
.           { is_valid = 0; /* Invalid character */ }

%%

int main() {
    printf("Enter an arithmetic expression: ");
    yylex();
    if (is_valid) {
        printf("The expression is valid.\n");
            printf("Identifiers: %d\n", identifier_count);
            printf("Operators: %d\n", operator_count);

    } else {
        printf("The expression is not valid.\n");
    }
    return 0;
}

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

```
Input: a+b*c
Output: The expression is not valid.

Input: 1*2+3
Output: The expression is valid.
Identifiers: 3
Operators: 2
```

Problem Statement:

2) Write YACC program to evaluate arithmetic expression involving operators: +, -, *, and /.

```
%{
        /* Definition section*/
        #include "y.tab.h"
        extern yylval;
```

```
%}


%%
[0-9]+ {
                        yylval = atoi(yytext);
                        return NUMBER;
                        }

[a-zA-Z]+ { return ID; }
[ \t]+          ; /*For skipping whitespaces*/

\n              { return 0; }
.               { return yytext[0]; }

%%

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

%token NUMBER ID
// setting the precedence
// and associativity of operators
%left '+' '-'
%left '*' '/'

/* Rule Section */
%%
E : T     {
```

```
                                        printf("Result = %d\n", $$);
                                        return 0;
                              }

T :
          T '+' T { $$ = $1 + $3; }
          | T '-' T { $$ = $1 - $3; }
          | T '*' T { $$ = $1 * $3; }
          | T '/' T { $$ = $1 / $3; }
          | '-' NUMBER { $$ = -$2; }
          | '-' ID { $$ = -$2; }
          | '(' T ')' { $$ = $2; }
          | NUMBER { $$ = $1; }
          | ID { $$ = $1; };
% %

int main() {
          printf("Enter the expression\n");
          yyparse();
}

/* For printing error messages */
int yyerror(char* s) {
          printf("\nExpression is invalid\n");
}
```

Input: 7*(5-3)/2
Output: 7

Input: 6/((3-2)*(-5+2))
Output: -2

Problem Statement:

3) Develop, Implement and execute a program using YACC tool to recognize all strings ending with b preceded by n a's using the grammar a n b (note: input n value).

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

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

int yywrap()
{
return 1;
}

%{
/* 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();
}
```

```
Input: ab
Output: valid string

Input: aba
Output: invalid string
```

Problem Statement:

4) Write a LEX program to eliminate comment lines in a C program and copy the resulting program into a separate file.

```
/% Lex Program to remove comments from C program
and save it in a file %/
/*Definition Section*/
%{
%}

/*Starting character sequence for multiline comment*/
start \/\*
/*Ending character sequence for multiline comment*/
end \*\/

/*Rule Section*/
%%

/*Regular expression for single line comment*/
\/\/(.*) ;
/*Regular expression for multi line comment*/
{start}.*{end} ;

%%

/*Driver function*/
int main(int k,char **argcv)
{
yyin=fopen(argcv[1],"r");
yyout=fopen("out.c","w");
/*call the yylex function.*/
yylex();

return 0;

}
```

Output:

```
Input :
//testing
#include
int main()
{
 /* multiline comment continue....
 */
 return 0;
}

Output :
#include
int main()
{

 return 0;
}
```

Problem Statement:

5) Write YACC program to recognize valid identifier, operators and keywords in the given text (C program) file.

```
%{

#include <stdio.h>

#include "y.tab.h"

extern yylval;

%}

%%

[ \t];

[+|-|*|/|=|<|>] {printf("operator is %s\n",yytext);return OP;}
```

```
[0-9]+ {yylval = atoi(yytext); printf("numbers is %d\n",yylval); return DIGIT;}
int|char|bool|float|void|for|do|while|if|else|return|void {printf("keyword
is %s\ n",yytext);return KEY;}
[a-zA-Z0-9]+ {printf("identifier is %s\n",yytext);return ID;}
. ;
%%

%{
#include <stdio.h>
#include <stdlib.h>
int id=0, dig=0, key=0, op=0;
%}
%token DIGIT ID KEY OP
%%
input:
DIGIT input { dig++; }
| ID input { id++; }
| KEY input { key++; }
| OP input {op++;}
| DIGIT { dig++; }
| ID { id++; }
| KEY { key++; }
| OP { op++;}
;
%%
#include <stdio.h>
extern int yylex();
extern int yyparse();
extern FILE *yyin;
main()
```

```
{
FILE *myfile = fopen("f2.c", "r");
if (!myfile)
{
printf("I can't open f2.c!");
return -1;
}
yyin = myfile;
do{  yyparse(
);
}while (!feof(yyin));
printf("numbers = %d\nKeywords = %d\nIdentifiers = %d\noperators = %d\n",dig, key,id,
op);
}
void yyerror() {
printf("EEK, parse error! Message: ");
exit(-1);
}
```

Input:

```
int main()
   { int a = 5;
   float b = 3.14;
   if (a == b) {
      return 0;
   } else {
      return 1;
   }
}
```

Output:

```
Keyword: int
```

```
Identifier: main
Operator:
( Operator: )
Operator:
{ Keyword: int
Identifier: a
Operator: =
Identifier: 5
Operator: ;
Keyword: float
Identifier: b
Operator: =
Identifier: 3.14
Operator: ;
Keyword: if
Operator:
( Identifier: a
Operator: ==
Identifier: b
Operator: )
Operator:
{ Keyword: return
Identifier: 0
Operator: ;
Operator: }
Keyword: else
Operator:
{ Keyword: return
Identifier: 1
Operator: ;
```

Problem Statement:

6)Write LEX program to count the number of characters, words, spaces and lines in each input file.

Solution:

```
/* DESCRIPTION/DEFINITION SECTION */
%{
#include<stdio.h>
int lc=0,sc=0,tc=0,ch=0,wc=0;  // GLOBAL VARIABLES
%}

// RULE SECTION
%%
[\n]  {  lc++;  ch+=yyleng;}
[ \t] { sc++; ch+=yyleng;}
[^\t] { tc++; ch+=yyleng;}
[^\t\n ]+ { wc++; ch+=yyleng;}
%%

int yywrap(){ return 1; }
/*          After inputting press ctrl+d                  */

// MAIN FUNCTION
int main(){
            printf("Enter the Sentence : ");
            yylex();
            printf("Number of lines : %d\n",lc);
            printf("Number of spaces : %d\n",sc);
            printf("Number of tabs, words, charc : %d , %d , %d\n",tc,wc,ch);
            return 0;
}
```

Input:
```
Hello
How    are    you?
```

Output:
```
Number of lines : 2
Number of spaces : 8
Number of tabs, words, charc : 0 , 4 , 25
```

Problem Statement:

7) Write a LEX program to recognize whether a given sentence is simple or compound.

Solution:

```
%{
#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;
}
```

Output:

```
Enter the sentence:
Hi
Simple sentence

Enter the sentence:
Hi and Hey
Compound sentence
```

Problem Statement:

8) Write a YACC program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.

```
%{

   #include "y.tab.h"

%}

%%

[a-zA-Z_][a-zA-Z_0-9]* return letter;

[0-9]                  return digit;

.                  return yytext[0];

\n                 return 0;

%%

int yywrap()

{

return 1;

}
```

Input: Enter a name to tested for identifier : abc

Output: It’s an identifier