1. a) 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.

%{

int id=0, num=0, op=0, top=-1, i;

char stack[10], ident[10][10], opr[10];

%}

id [\_a-zA-Z][\_a-zA-Z0-9]\*

num [0-9]+

%%

{id} { strcpy(ident[id], yytext);

id++;

}

{num} { num++; }

"+" { opr[op++]='+'; }

"-" { opr[op++]='-'; }

"\*" { opr[op++]='\*'; }

"/" { opr[op++]='/'; }

"(" { stack[++top]='('; }

")" { if(stack[top] == '(' && top!=-1)

top--;

else

{ printf("invalid expression\n");

exit(0);

}

}

"\n" { return; }

. { printf("invalid expression(it has special character)\n");

exit(0); }

%%

int main()

{

printf("Enter the expression :");

yylex();

if(top==-1 && id+num==op+1)

{

printf("valid expression\n");

printf("\n number of identifiers=%d and are \n",id);

for(i=0;i<id;i++)

printf("%s\n",ident[i]);

printf("\n number of operator=%d and are \n",op);

for(i=0;i<op;i++)

printf("%c\n",opr[i]);

}

else

printf("invalid expression\n"); }

Output 1:

[exam@localhost ~]$ lex 1a.l

[exam@localhost ~]$ cc lex.yy.c –ll

[exam@localhost ~]$ ./a.out

Enter the expression :(a+b)/(c\*d)-e

valid expression

number of identifiers=5 and are a b c d e

number of operator=4 and are + / \* -

Output 2:

[exam@localhost ~]$ ./a.out

Enter the expression :(a+b

invalid expression

b) Write YACC program to evaluate arithmetic expression involving operators: +, -, \*, and

/\* Lex part 1b.l\*/

%{

#include "y.tab.h"

extern int yylval;

%}

num [0-9]+

%%

{num} { yylval=atoi(yytext);

return NUM;

}

[\t] ;

. {return yytext[0];}

\n {return 0;}

%%

/\* Yacc part 1b.y\*/

%{

#include<stdio.h>

int result;

%}

%token NUM

%left '+' '-'

%left '\*' '/'

%%

S : E {result=$1;}

E :E '+' E {$$ = $1 + $3;}

|E '-' E {$$ = $1 - $3;}

|E '\*' E {$$ = $1 \* $3;}

|E '/' E {

if($3!=0)$$ = $1/$3;

else { printf("divide by zero error\n");

exit(0);

}

}

|'(' E ')' {$$ = $2;}

|NUM {$$ = $1;}

%%

int main()

{

printf("enter the expression\n");

yyparse();

printf("valid expression\n");

printf("result.....:%d\n",result);

}

yyerror()

{

printf("invalid expression\n");

exit(0);

}

Output 1:

[exam@localhost ~]$ lex 1b.l

[exam@localhost ~]$ yacc -d 1b.y

[exam@localhost ~]$ cc lex.yy.c y.tab.c -ll

[exam@localhost ~]$ ./a.out

enter the expression (4+5)\*(3-1)

valid expression

result.....:18

Output 2: [exam@localhost ~]$ ./a.out

enter the expression 5+

invalid expression

2. Develop, Implement and Execute a program using YACC tool to recognize all strings ending with b preceded by n a’s using the grammar an b (note: input n value

/\* Lex part 2.l\*/

%{

#include "y.tab.h"

%}

%%

a {return A;}

b {return B;}

[\t] ;

. {return yytext[0];}

\n {return 0;}

%%

/\* Yacc part 2.y\*/

%{

#include <stdio.h>

%}

%token A B

%%

S: X B

X: A X

| ;

%%

int main()

{

printf("\nEnter the string\n");

yyparse();

printf("String is valid\n");

}

yyerror()

{

printf("String invalid\n");

exit(0);

}

Output 1: [exam@localhost ~]$ lex 2.l

[exam@localhost ~]$ yacc -d 2.y

[exam@localhost ~]$ cc lex.yy.c y.tab.c –ll

[exam@localhost ~]$ ./a.out

Enter the string aaaaaaaaaaab

String is valid

Output 2: [exam@localhost ~]$ ./a.out

Enter the string ab String invalid

Output 3: [exam@localhost ~]$ ./a.out

Enter the string baaaaaaaaaaa

String invalid

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

%{

int comment=0, state=1;

%}

%%

"/\*" { state=0; }

"\*/" { if(state==0)

{ comment++;

state=1;

}

}

.|\n { if(state==1) /\* not a comment line\*/

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

}

%%

int main()

{

char infile[256],outfile[256];

printf("Enter the input filename : ");

scanf("%s",infile);

printf("Enter the output filename : ");

scanf("%s",outfile);

yyin=fopen(infile,"r");

yyout=fopen(outfile,"w");

yylex();

printf("\n number of comment lines in the given file: %d\n",comment);

}

Output: [exam@localhost ~]$ vi a.c

/\*program to add two nos\*/

main()

{

int a,b,c;

printf("Enter two nos:");

scanf("%d%d",&a,&b);/\*read a and b\*/

c=a+b; /\*adition\*/

printf("Sum=%d",c);/\*display sum\*/

}

[exam@localhost ~]$ lex 6a.l

[exam@localhost ~]$ cc lex.yy.c –ll

[exam@localhost ~]$ ./a.out

Enter the input filename : a.c

Enter the output filename : b.c

number of comment lines in the given file: 4

[exam@localhost ~]$ cat b.c

main() {

int a,b,c;

printf("Enter two nos:");

scanf("%d%d",&a,&b);

c=a+b;

printf("Sum=%d",c);

}

b) 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;}

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;}

. ;

%%

Yacc File

%{

#include <stdio.h>

#include <stdlib.h>

int id=0, dig=0, key=0, op=0;

%}

%token ID KEY OP

%%

input: DIGIT input { dig++; }

| ID input { id++; }

| KEY input { key++; }

| OP input {op++;}

| ID { id++; }

| KEY { key++; }

| OP { op++;}

;

%%

#include <stdio.h>

extern int yylex();

extern int yyparse();

extern FILE \*yyin;

main() {

FILE \*myfile = fopen("sam\_input.c", "r");

if (!myfile) { printf("I can't open sam\_input.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("parse error! Message: ");

exit(-1);

}