System Software (CS306)

LAB TEST 1

U20CS135

1. Write a Lex Program.

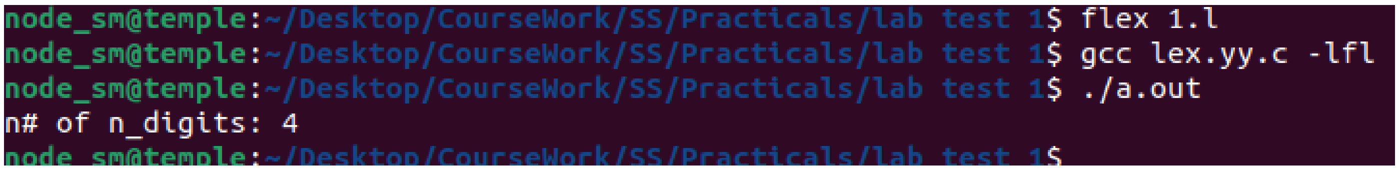
Input: 1234 Output: Number of digits = 4

```
%{ #include<studio.h>

int n_digits=0;

%}


%%

[0-9] {++n_digits;}

. printf("Invalid");


%%
```

```
int main(int argc[],char *argv[])
{
 yyin=fopen("shivam.txt", "r");
 yylex();
 printf("n# of n_digits:
%d",n_digits);
 printf("\n");


 return 0;
}
```

shivam.txt

```
1234
```



```
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ flex 1.l
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ gcc lex.yy.c -lfl
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ ./a.out
n# of n_digits: 4
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$
```

2. Write a Lex Program.

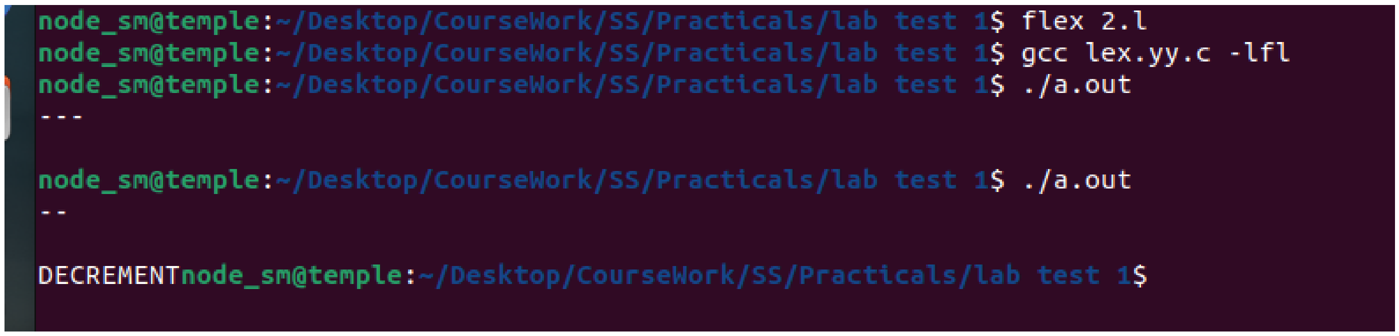
Input: - Output: MINUS

Input: - - Output: DECREMENT

Input: - - - Output: DECREMENT MINUS

```
%{ #include<studio.h>

int cnt=0;

%}



%%
```

```
"-" cnt++;

. printf("Invalid");

%%

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

 yylex();
 if(cnt==3)

 printf("DECREMENT MINUS");

 else if(cnt==2)

 printf("DECREMENT");

 else if(cnt==1)

 printf("MINUS");




 return 0;

}
```



```
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ flex 2.l
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ gcc lex.yy.c -lfl
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ ./a.out
---

node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ ./a.out
--

DECREMENTnode_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$
```

3.3. Program to recognize a valid arithmetic expression and identify the identifiers

and operators present. Print them seperatly.

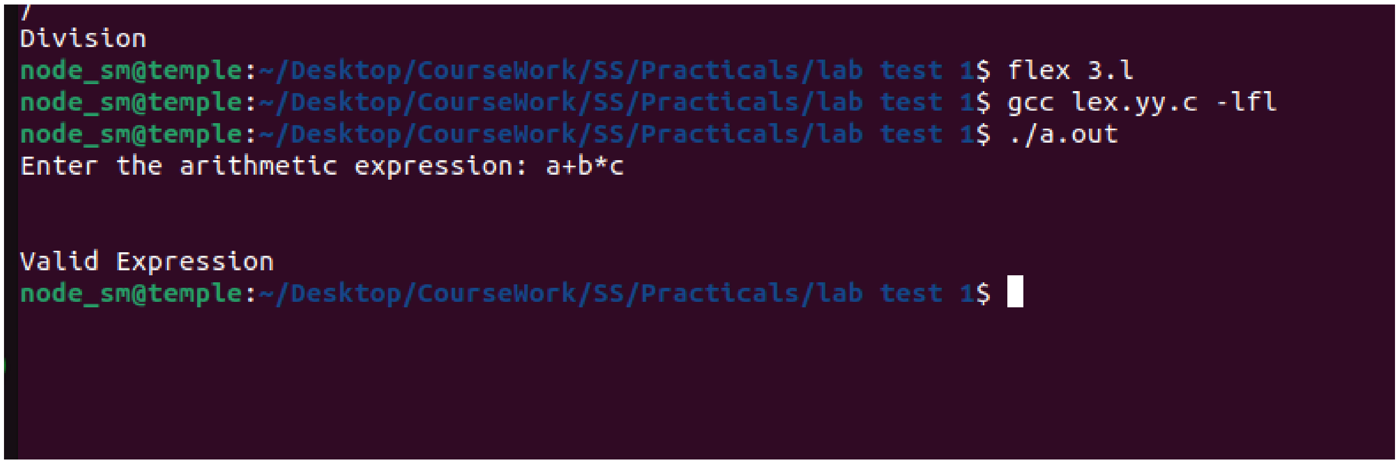
```
/* Lex program to recognize valid arithmetic expression

        and identify the identifiers and operators */

%{

#include <stdio.h>

#include <string.h>
```

```
    int operators_count = 0, operands_count = 0, valid = 1, top = -1, l = 0, j = 0;

    char operands[10][10], operators[10][10], stack[100];

%}

%%

"(" {

    top++;

    stack[top] = '(';

}

"{" {

    top++;
    stack[top] = '{';

}

"[" {

    top++;

    stack[top] = '[';

}

")" {

    if (stack[top] != '(') {

        valid = 0;

    }
    else if(operands_count>0 && (operands_count-operators_count)!=1){

        valid=0;

    }

    else{

        top--;

        operands_count=1;

        operators_count=0;

    }

}

"}" {
```

```
if (stack[top] != '{') {

    valid = 0;

}

else if(operands_count>0 && (operands_count-operators_count)!=1){

    valid=0;

}

else{

    top--;

    operands_count=1;

    operators_count=0;
    }

}

"]" {

    if (stack[top] != '[') {

        valid = 0;

    }

    else if(operands_count>0 && (operands_count-operators_count)!=1){

        valid=0;

    }

    else{
        top--;

        operands_count=1;

        operators_count=0;

    }




}

"+"|"-"|"*"|"/" {

    operators_count++;

    strcpy(operators[l], yytext);

    l++;
```

```
}

[0-9]+|[a-zA-Z][a-zA-Z0-9_]* {

    operands_count++;

    strcpy(operands[j], yytext);

    j++;

}

%%




int yywrap()
```

```
{

    return 1;

}

int main()

{

    int k;

    printf("Enter the arithmetic expression: ");

    yylex();



    if (valid == 1 && top == -1) {
        printf("\nValid Expression\n");

    }

    else

        printf("\nInvalid Expression\n");



    return 0;

}
```



```
/
Division
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ flex 3.l
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ gcc lex.yy.c -lfl
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ ./a.out
Enter the arithmetic expression: a+b*c


Valid Expression
node_sm@temple:~/Desktop/CourseWork/SS/Practicals/lab test 1$ 
```

SUBMITTED BY:

U20CS135

Shivam Mishra