COMPILER DESIGN LAB FILE

Submitted by: Anshi

SAP ID:500101953

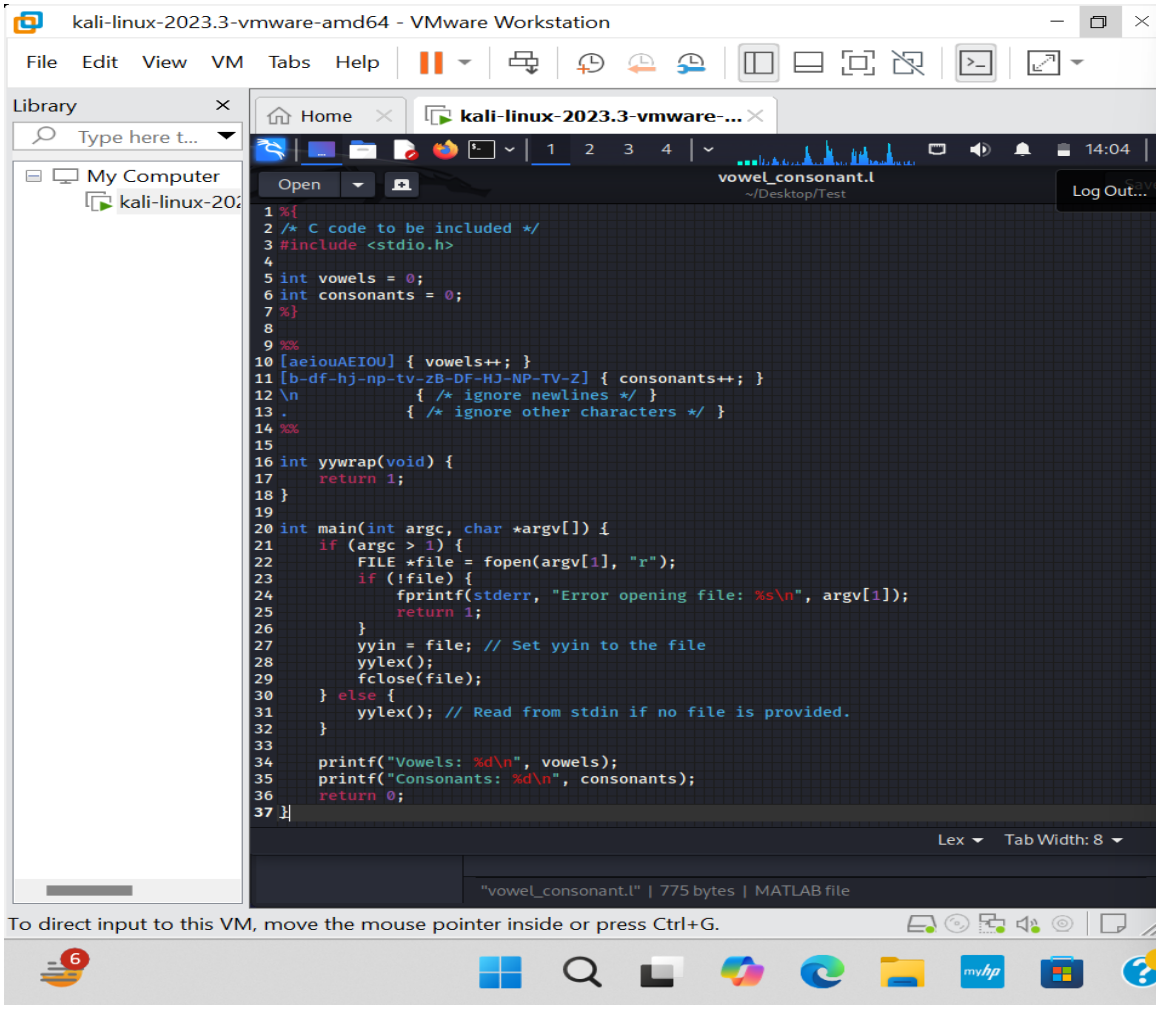
Batch: B4

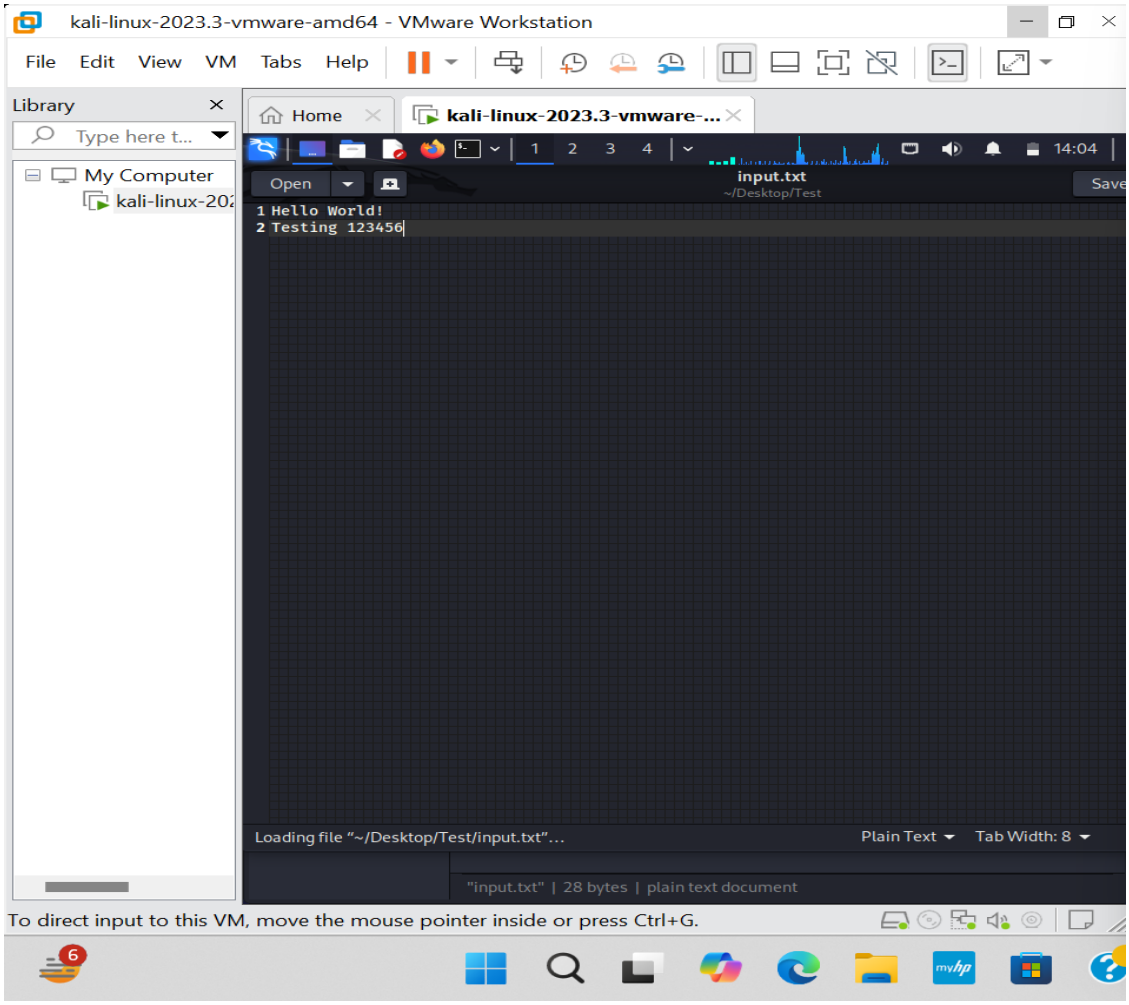
INDEX

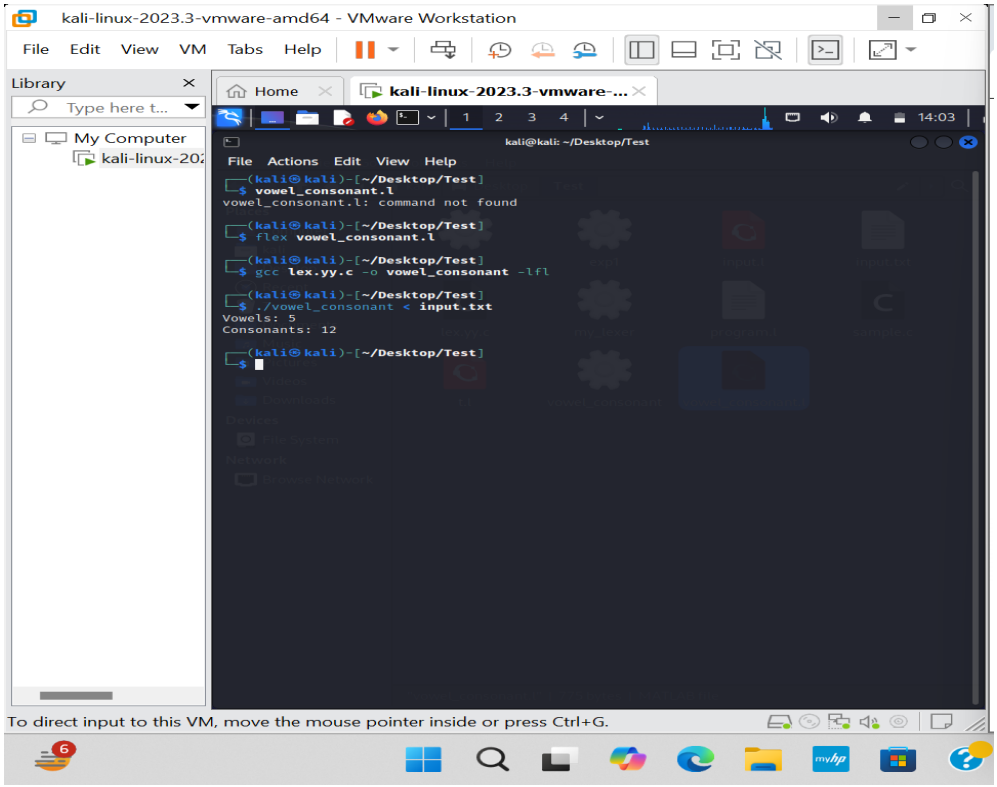
EXPERIMENT NO-2

Title: Introductory Problems using Lex Tools

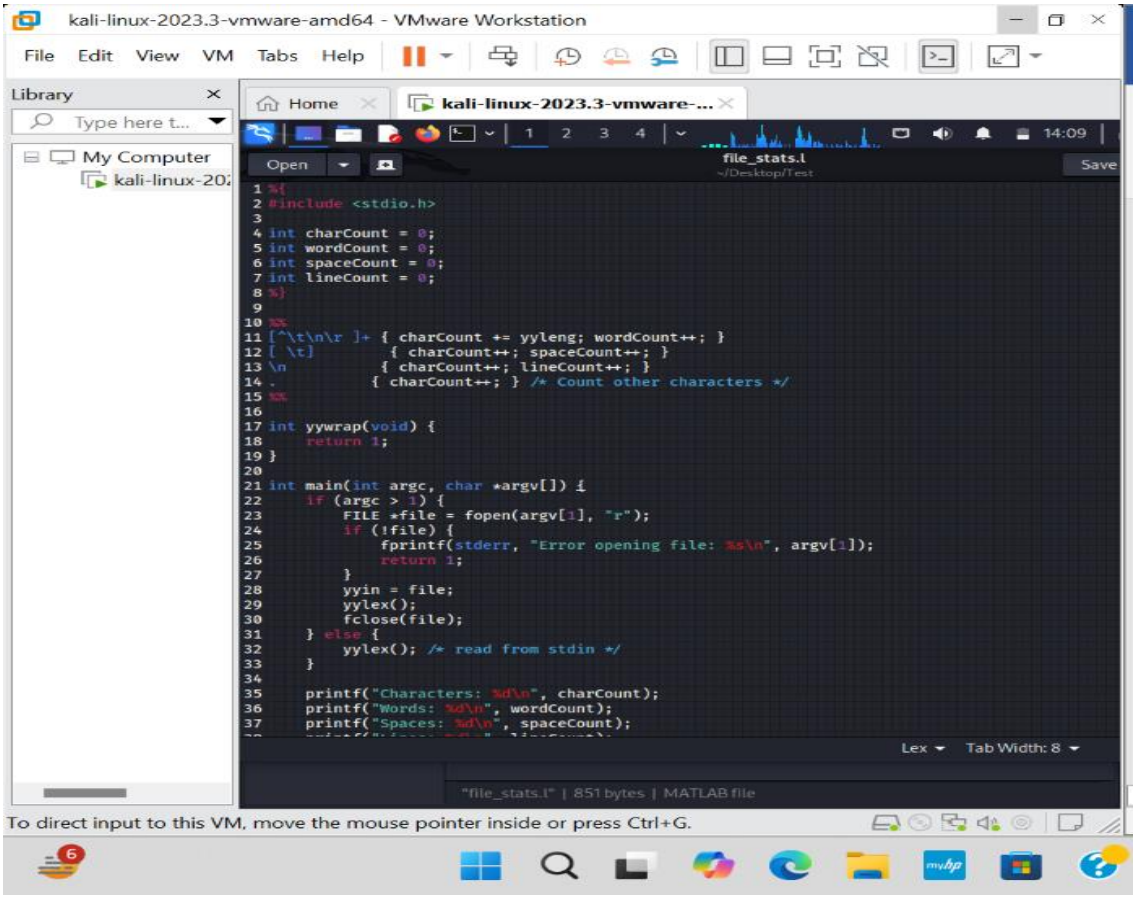
1. WAP to count number of vowels and consonants in a given string.

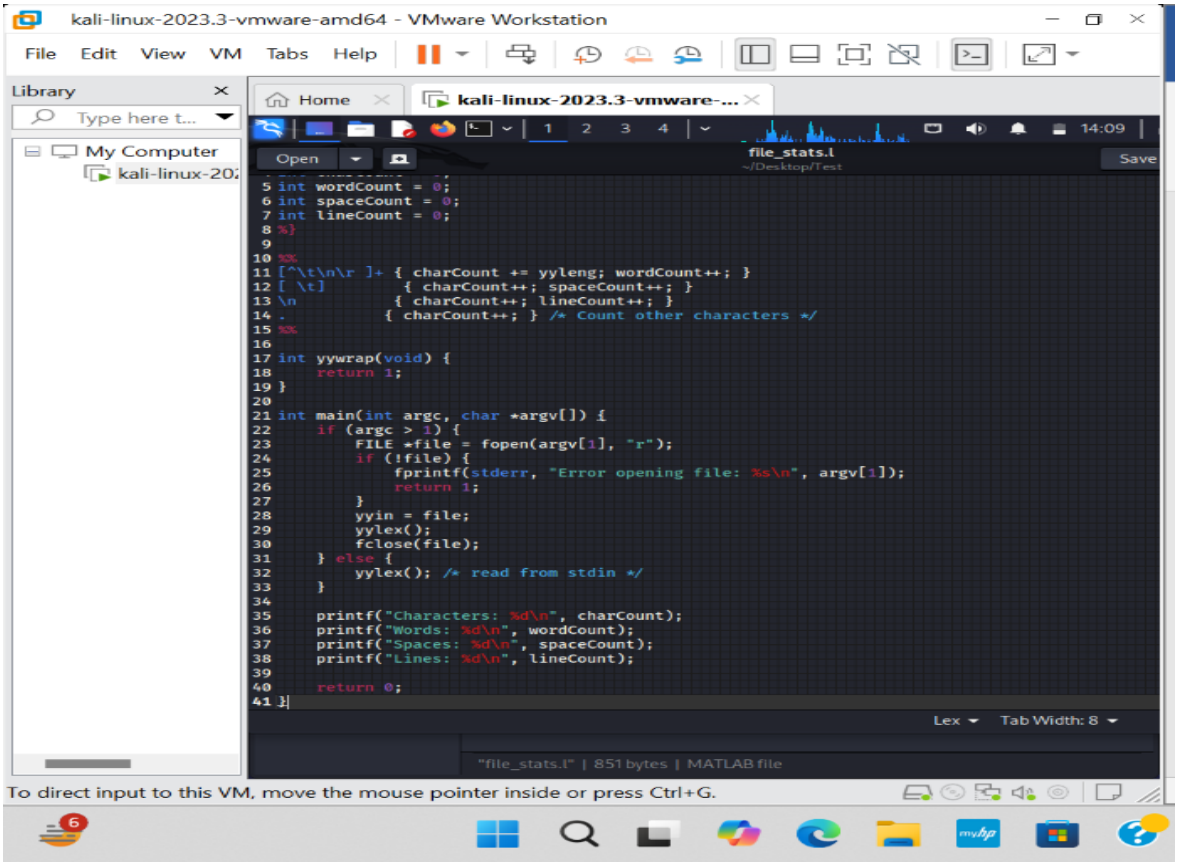


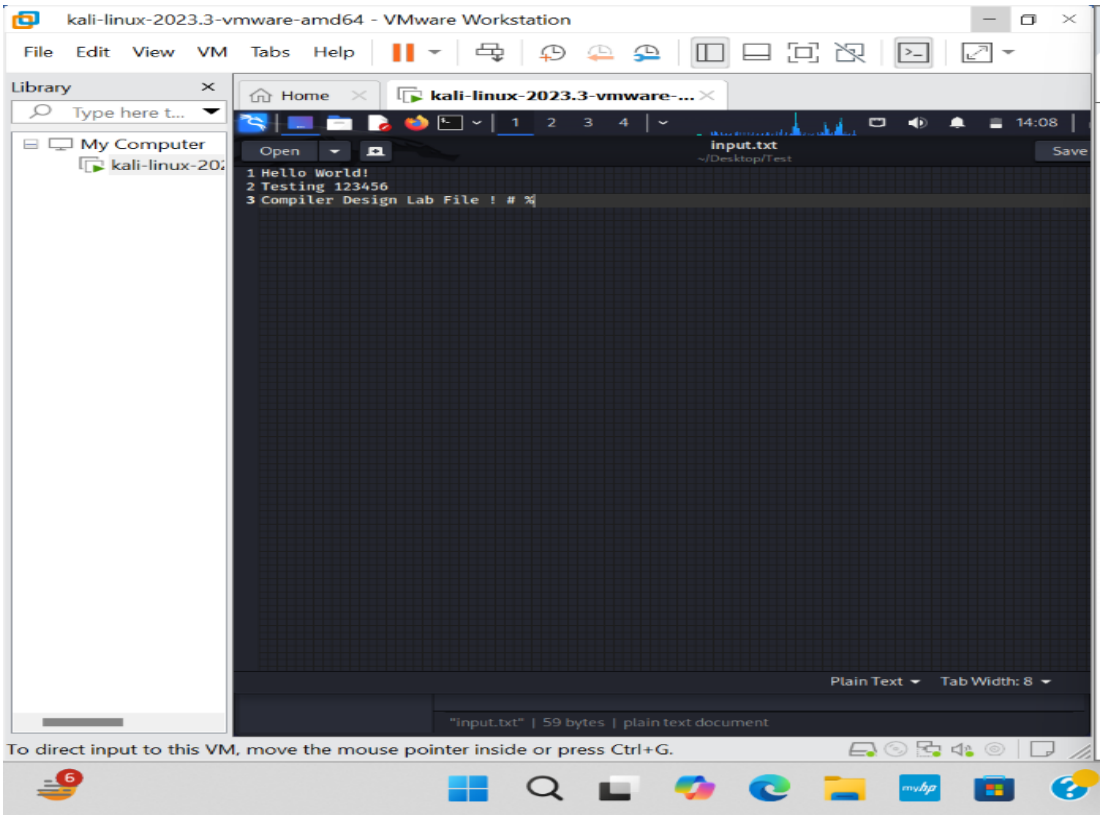

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
1 2 3 4
14:04
Open
input.txt
~/Desktop/Test
Save
1 Hello World!
2 Testing 123456
Loading file "~/Desktop/Test/input.txt"...
Plain Text
Tab Width: 8
"input.txt" | 28 bytes | plain text document
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

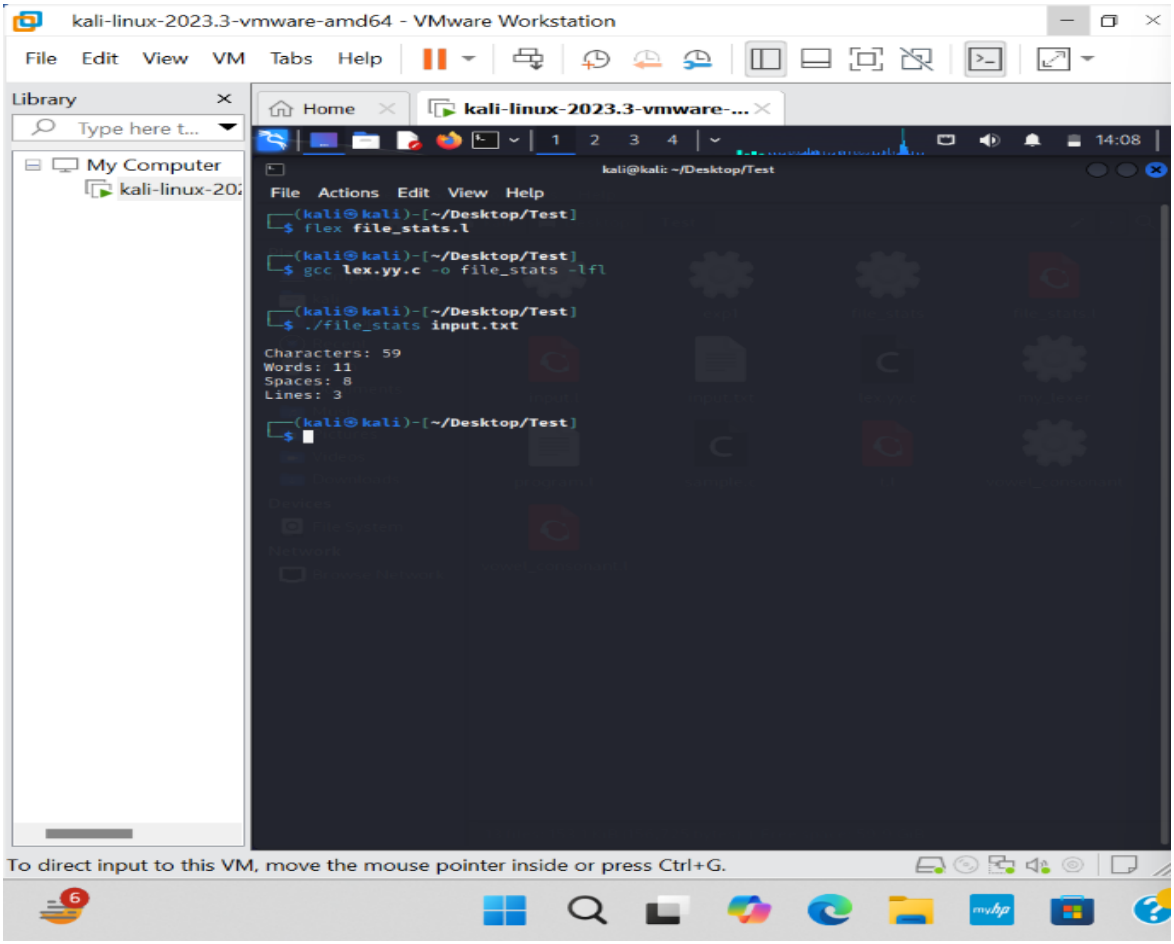

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
1 2 3 4
14:03
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ vowel_consonant.l
vowel_consonant.l: command not found
(kali㉿kali)-[~/Desktop/Test]
$ flex vowel_consonant.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o vowel_consonant -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./vowel_consonant < input.txt
Vowels: 5
Consonants: 12
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

2. WAP to count the number of characters, words, spaces, and end of lines in a given input file.

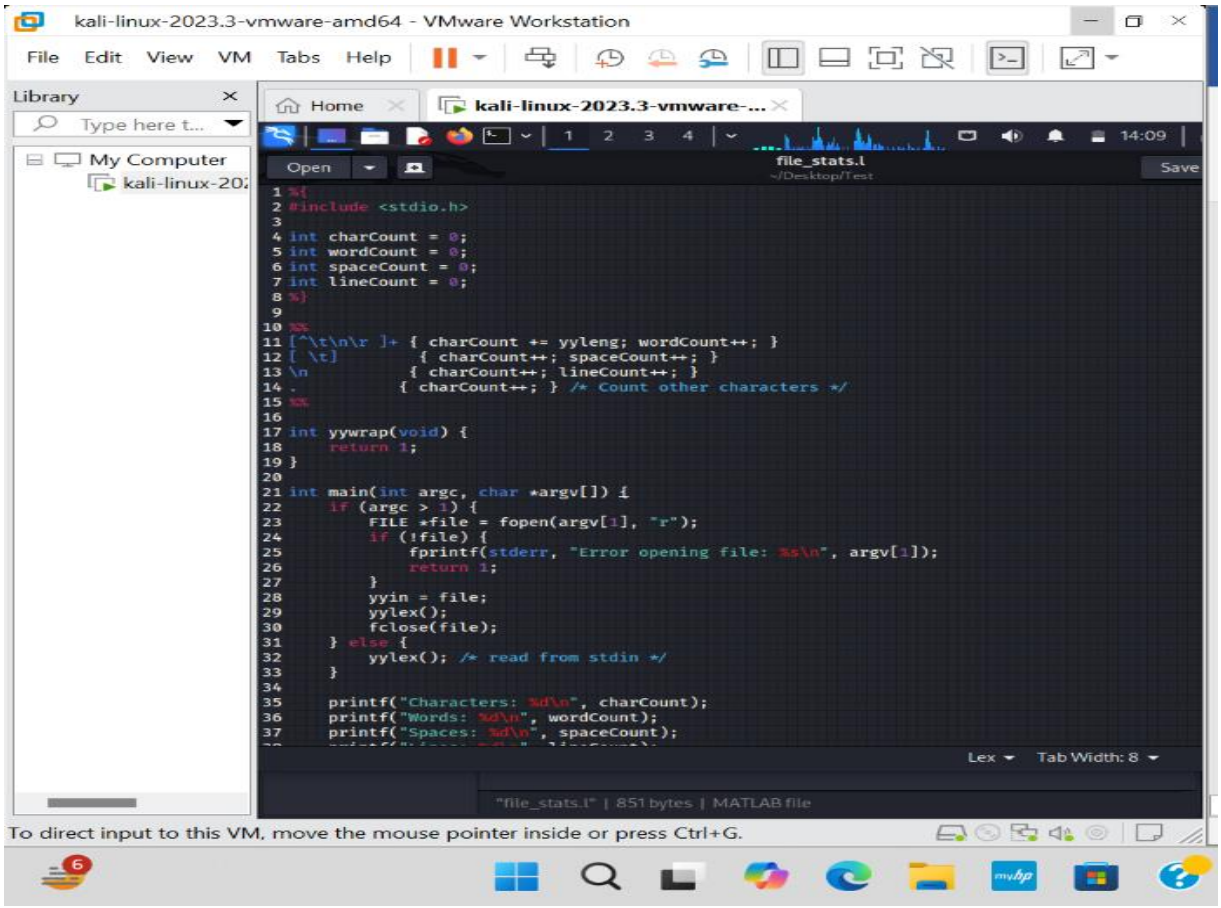


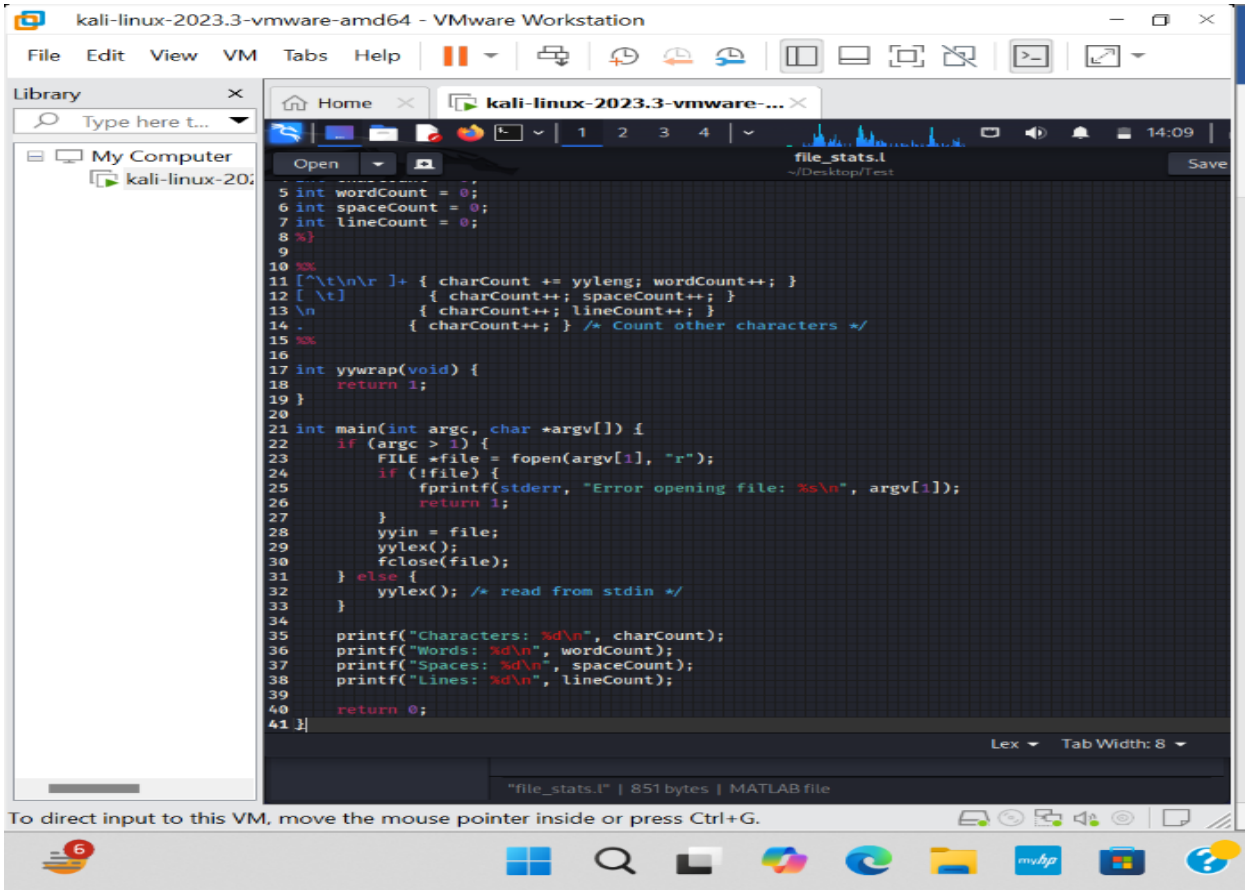

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:09
Open
file_stats.l
~/Desktop/Test
Save
5 int wordCount = 0;
6 int spaceCount = 0;
7 int lineCount = 0;
8 %}
9
10 %%
11 [^\t\n\r]+ { charCount += yyleng; wordCount++; }
12 [\t] { charCount++; spaceCount++; }
13 \n { charCount++; lineCount++; }
14 . { charCount++; } /* Count other characters */
15 %%
16
17 int yywrap(void) {
18 return 1;
19 }
20
21 int main(int argc, char *argv[]) {
22 if (argc > 1) {
23 FILE *file = fopen(argv[1], "r");
24 if (!file) {
25 fprintf(stderr, "Error opening file: %s\n", argv[1]);
26 return 1;
27 }
28 yyin = file;
29 yylex();
30 fclose(file);
31 } else {
32 yylex(); /* read from stdin */
33 }
34
35 printf("Characters: %d\n", charCount);
36 printf("Words: %d\n", wordCount);
37 printf("Spaces: %d\n", spaceCount);
38 printf("Lines: %d\n", lineCount);
39
40 return 0;
41 }
Lex
Tab Width: 8
"file_stats.l" | 851 bytes | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

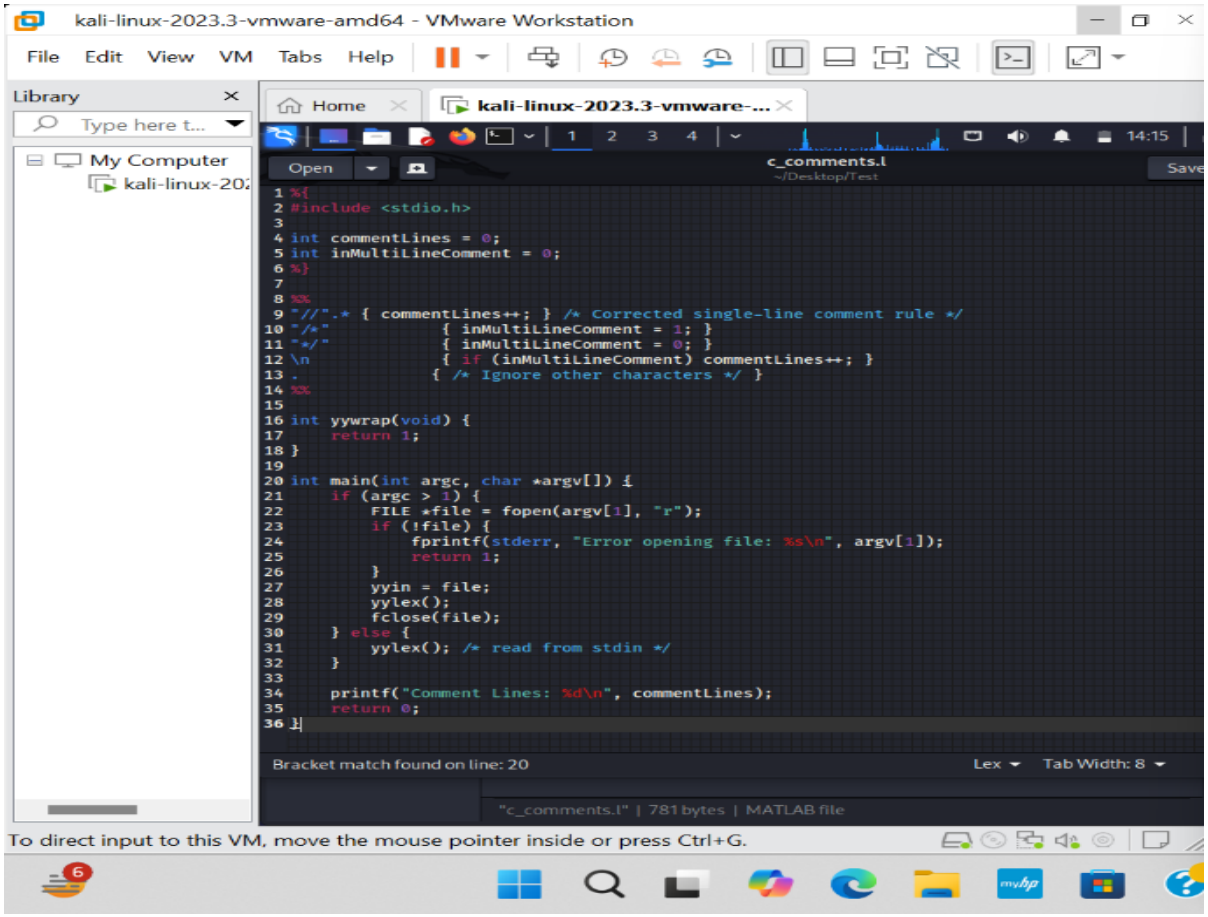

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:08
Open
input.txt
~/Desktop/Test
Save
1 Hello World!
2 Testing 123456
3 Compiler Design Lab File ! # %
Plain Text
Tab Width: 8
"input.txt" | 59 bytes | plain text document
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

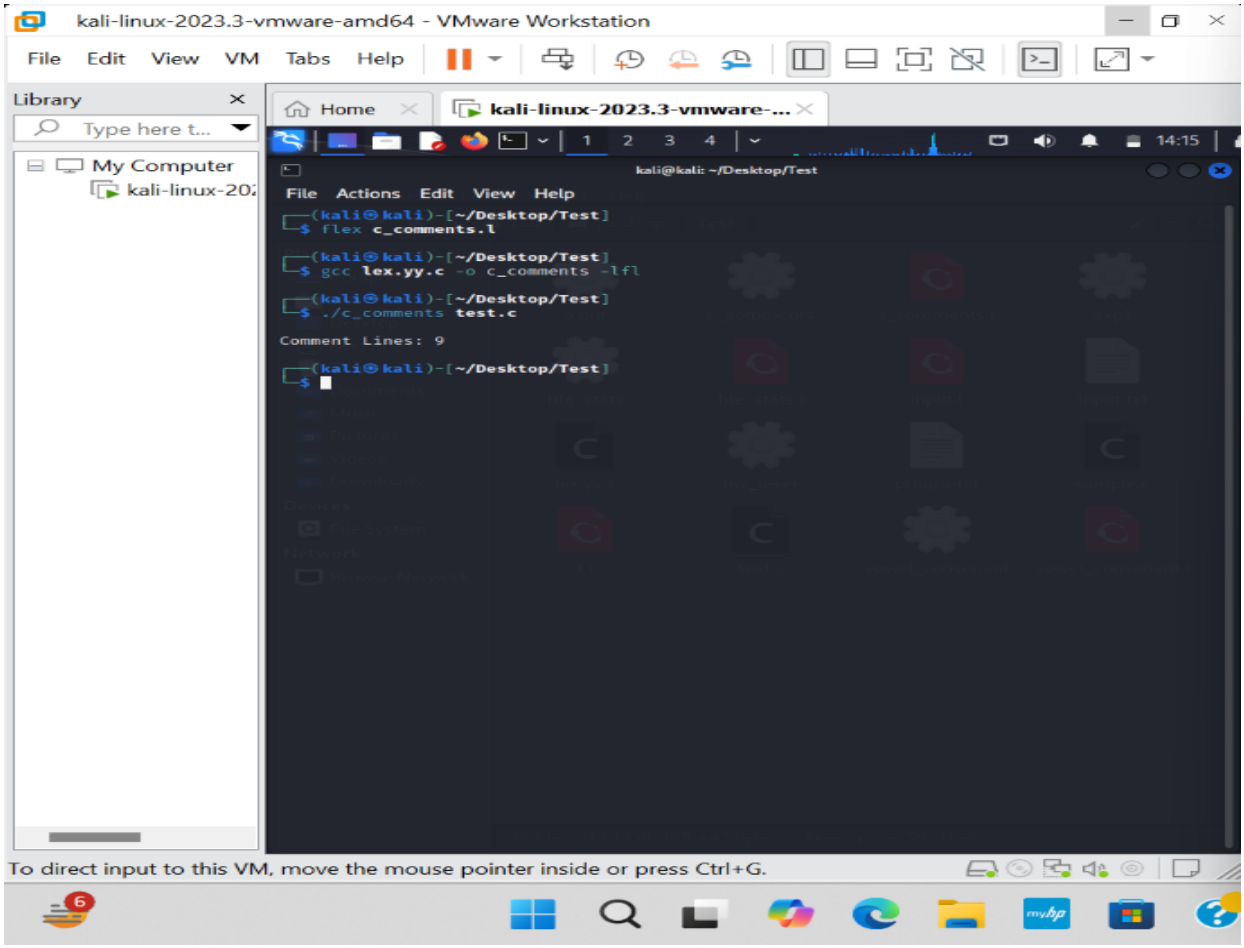

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:08
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex file_stats.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o file_stats -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./file_stats input.txt
Characters: 59
Words: 11
Spaces: 8
Lines: 3
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

3. WAP to count number of comment lines in a given C program.




kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:09
Open
file_stats.l
~/Desktop/Test
Save
5 int wordCount = 0;
6 int spaceCount = 0;
7 int lineCount = 0;
8 %}
9
10 %%
11 [^\t\n\r]+ { charCount += yyleng; wordCount++; }
12 [\t] { charCount++; spaceCount++; }
13 \n { charCount++; lineCount++; }
14 . { charCount++; } /* Count other characters */
15 %%
16
17 int yywrap(void) {
18 return 1;
19 }
20
21 int main(int argc, char *argv[]) {
22 if (argc > 1) {
23 FILE *file = fopen(argv[1], "r");
24 if (!file) {
25 fprintf(stderr, "Error opening file: %s\n", argv[1]);
26 return 1;
27 }
28 yyin = file;
29 yylex();
30 fclose(file);
31 } else {
32 yylex(); /* read from stdin */
33 }
34
35 printf("Characters: %d\n", charCount);
36 printf("Words: %d\n", wordCount);
37 printf("Spaces: %d\n", spaceCount);
38 printf("Lines: %d\n", lineCount);
39
40 return 0;
41 }
Lex
Tab Width: 8
"file_stats.l" | 851 bytes | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

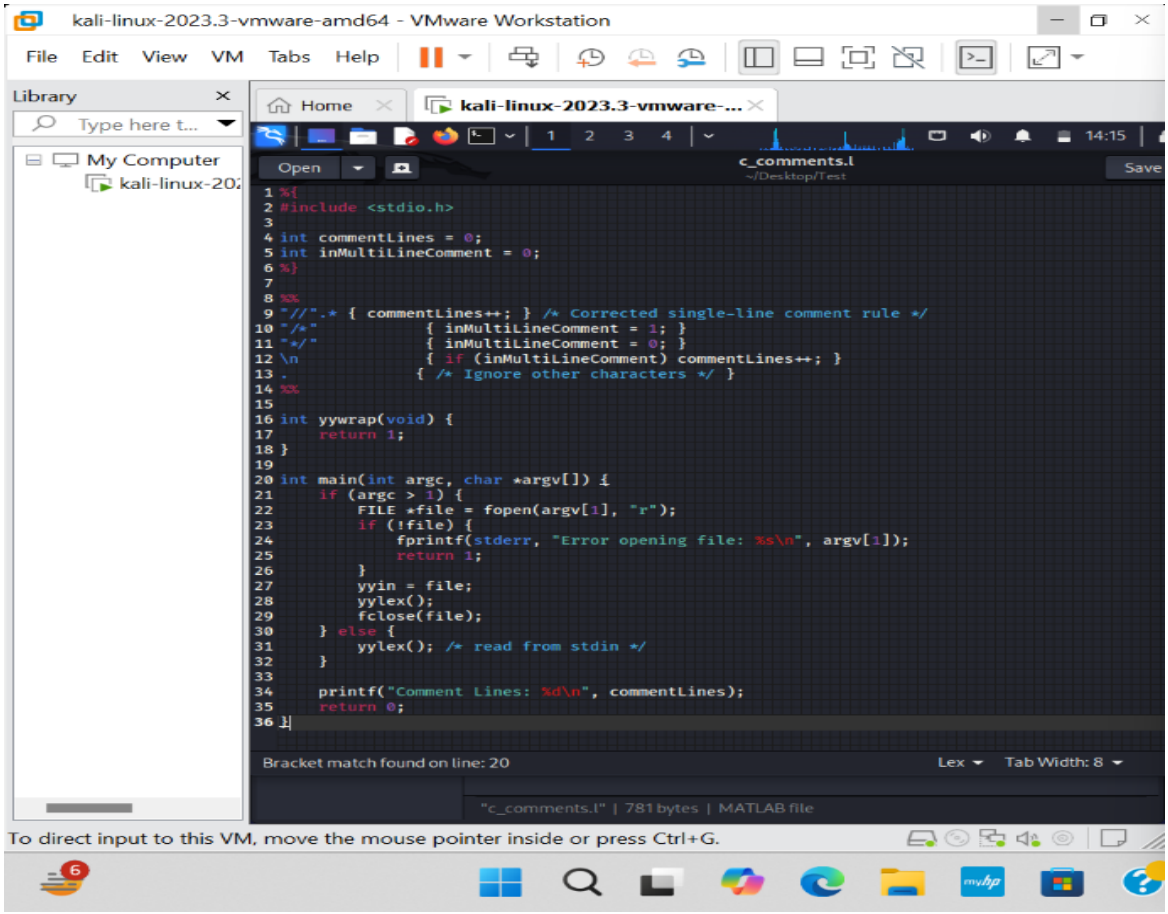

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
14:15
Open
c_comments.l
~/Desktop/Test
Save
%{
#include <stdio.h>
int commentLines = 0;
int inMultiLineComment = 0;
%}
%%
"//".* { commentLines++; } /* Corrected single-line comment rule */
"/*" { inMultiLineComment = 1; }
"*/" { inMultiLineComment = 0; }
\n { if (inMultiLineComment) commentLines++; }
. { /* Ignore other characters */ }
%%
int yywrap(void) {
return 1;
}
int main(int argc, char *argv[]) {
if (argc > 1) {
FILE *file = fopen(argv[1], "r");
if (!file) {
fprintf(stderr, "Error opening file: %s\n", argv[1]);
return 1;
}
yyin = file;
yylex();
fclose(file);
} else {
yylex(); /* read from stdin */
}
printf("Comment Lines: %d\n", commentLines);
return 0;
}
Bracket match found on line: 20
Lex
Tab Width: 8
"c_comments.l" | 781 bytes | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

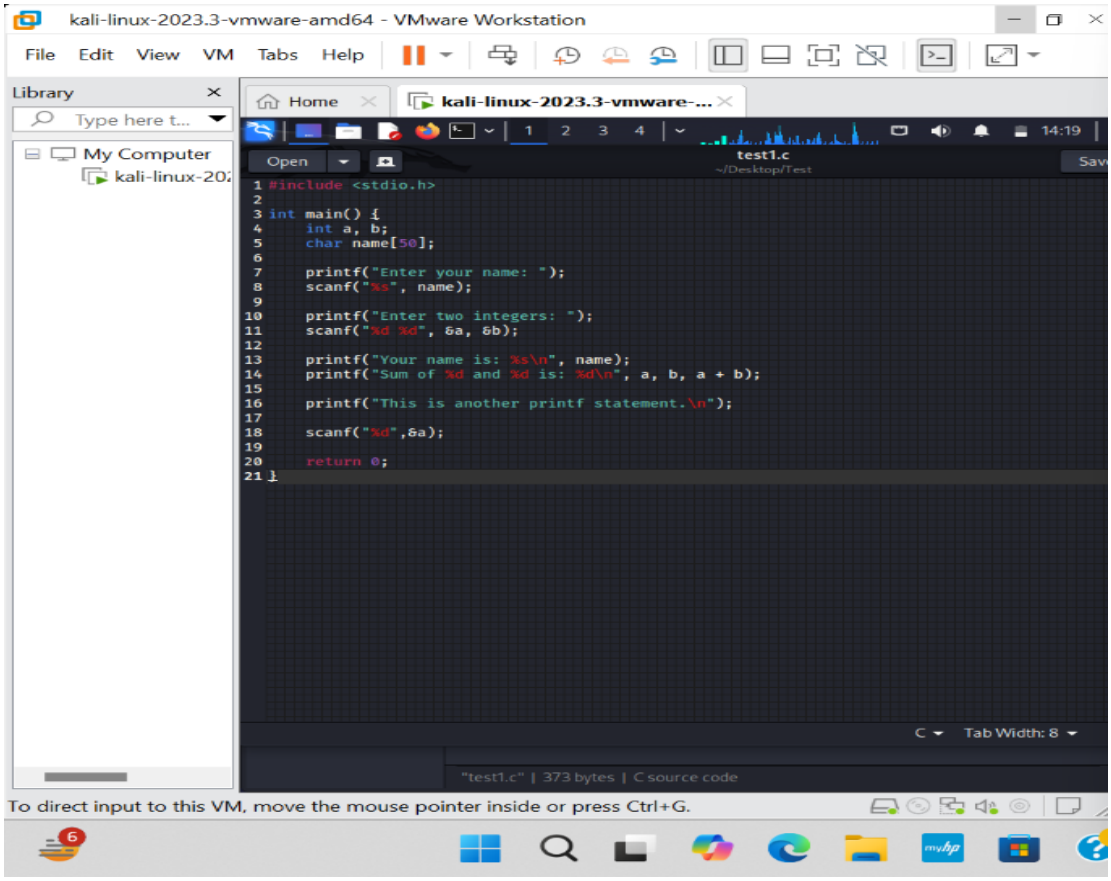

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex c_comments.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o c_comments -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./c_comments test.c
Comment Lines: 9
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

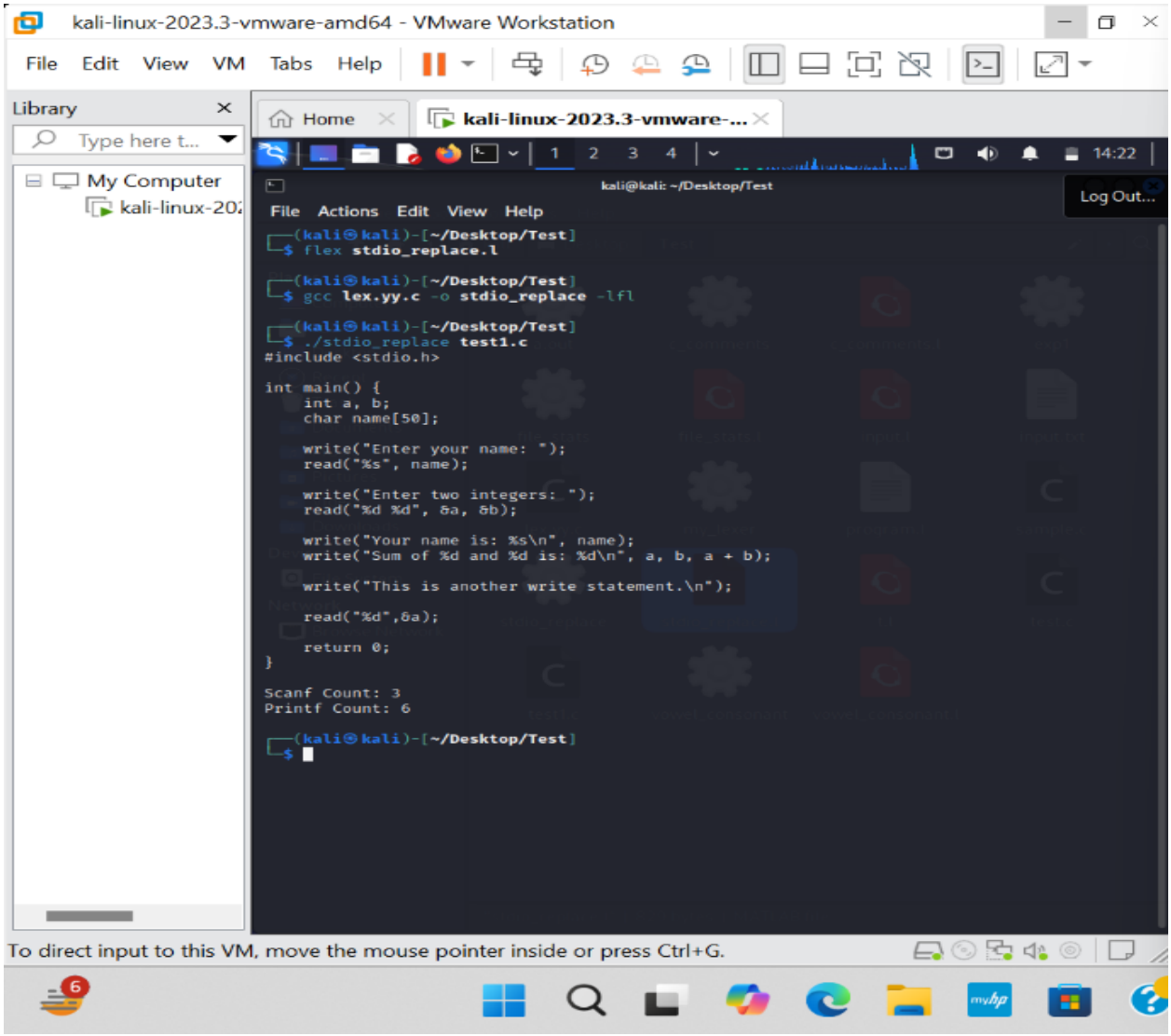
EXPERIMENT NO-3

Title: Lexical Analysis through Lexx Tools.

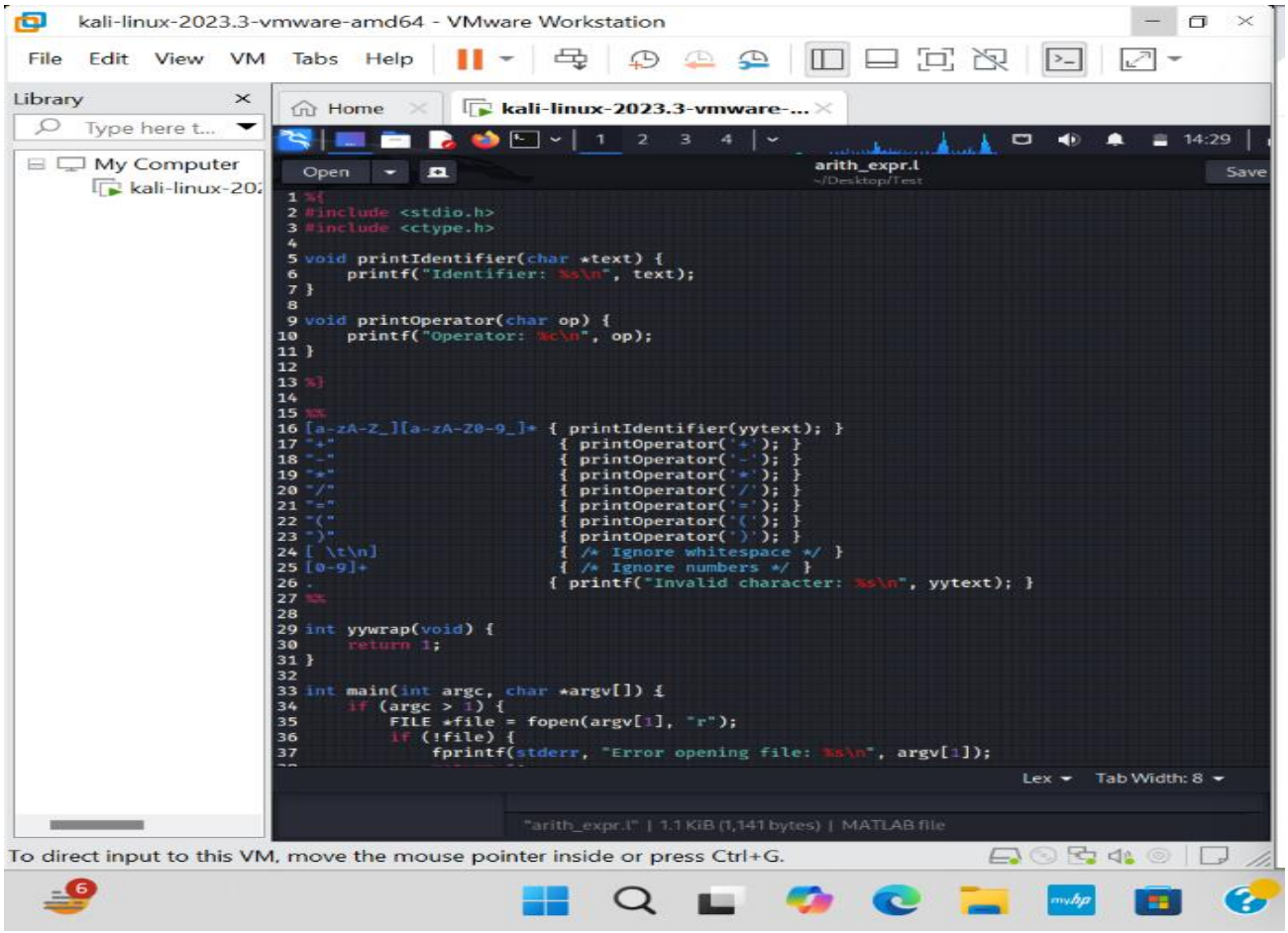
1. WAP to count number of ‘scanf’ and ‘printf’ statements in a C program. Replace them with ‘read’ and ‘write’ statements respectively.

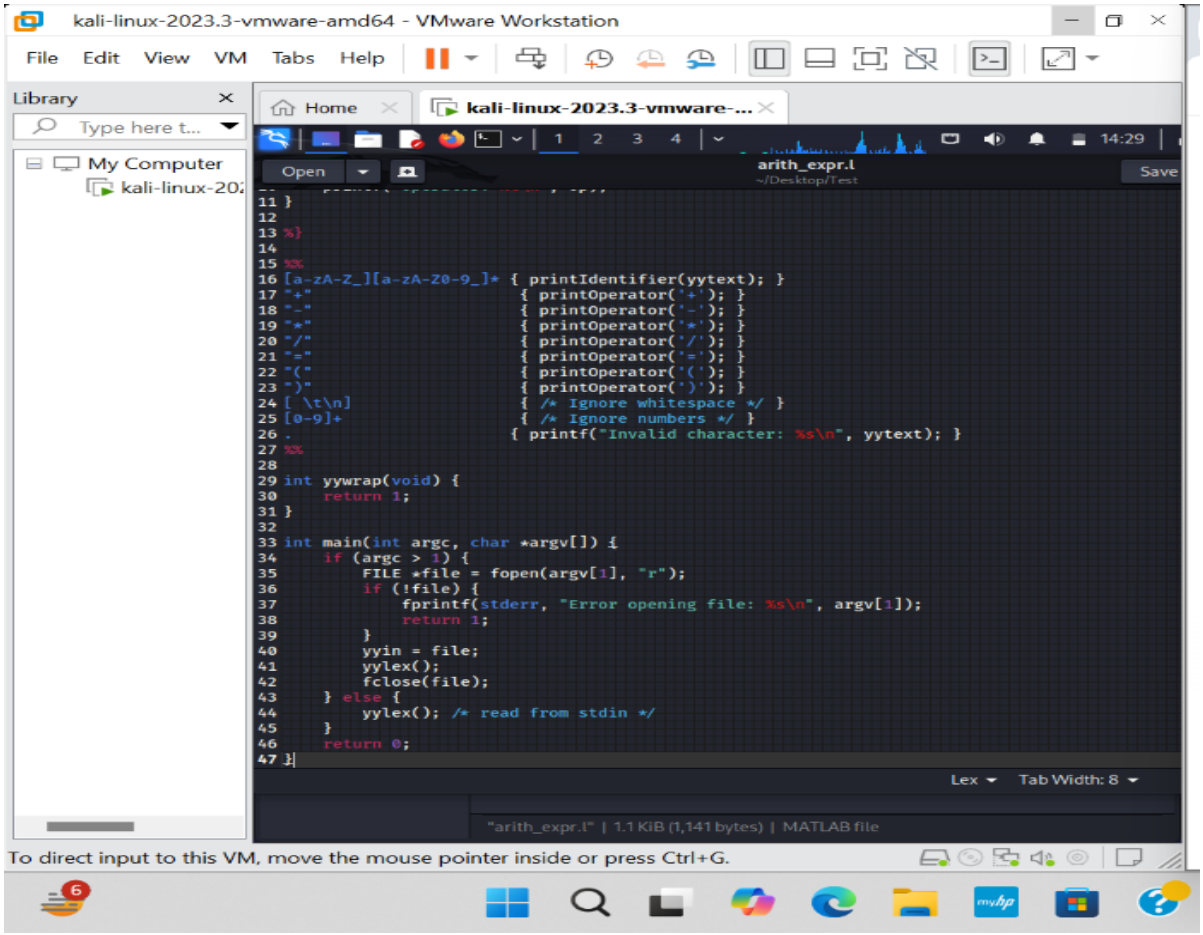


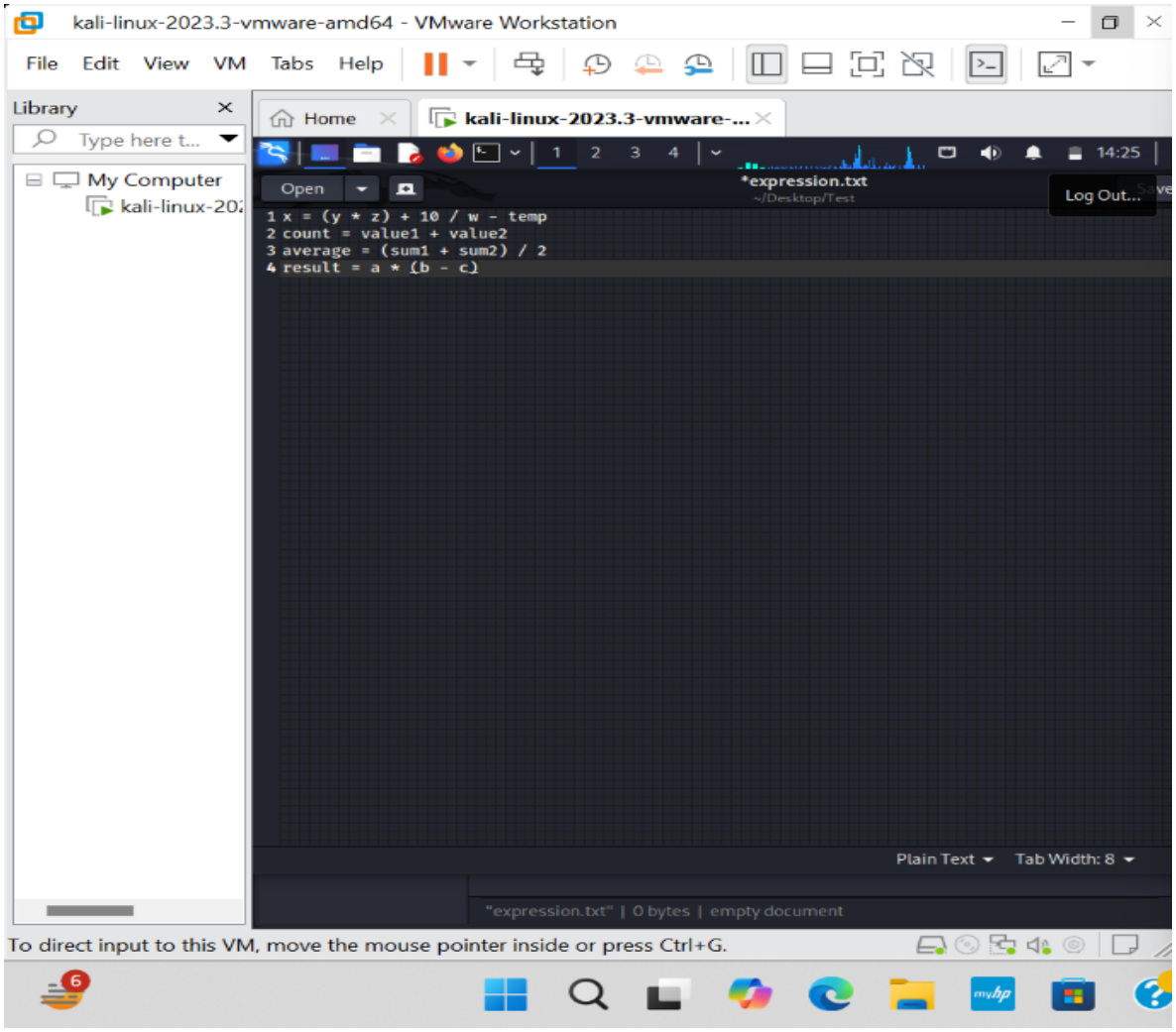

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
1 2 3 4
14:19
Open
test1.c
~/Desktop/Test
Sav
1 #include <stdio.h>
2
3 int main() {
4 int a, b;
5 char name[50];
6
7 printf("Enter your name: ");
8 scanf("%s", name);
9
10 printf("Enter two integers: ");
11 scanf("%d %d", &a, &b);
12
13 printf("Your name is: %s\n", name);
14 printf("Sum of %d and %d is: %d\n", a, b, a + b);
15
16 printf("This is another printf statement.\n");
17
18 scanf("%d",&a);
19
20 return 0;
21 }
C
Tab Width: 8
"test1.c" | 373 bytes | C source code
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

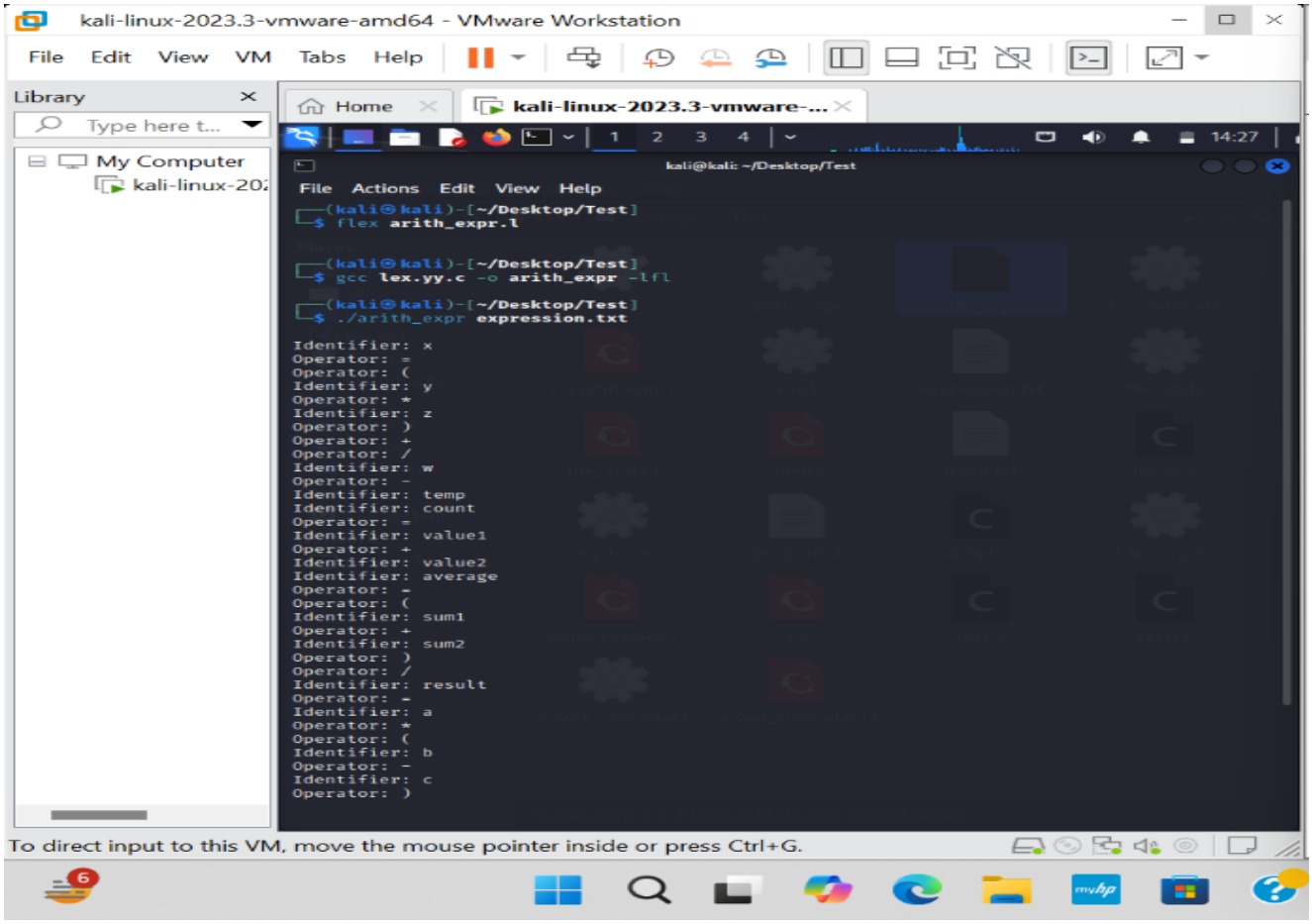

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
14:22
kali@kali: ~/Desktop/Test
Log Out...
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex stdio_replace.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o stdio_replace -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./stdio_replace test1.c
#include <stdio.h>
int main() {
int a, b;
char name[50];
write("Enter your name: ");
read("%s", name);
write("Enter two integers: ");
read("%d %d", &a, &b);
write("Your name is: %s\n", name);
write("Sum of %d and %d is: %d\n", a, b, a + b);
write("This is another write statement.\n");
read("%d",&a);
return 0;
}
Scanf Count: 3
Printf Count: 6
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

2. WAP to recognize a valid arithmetic expression and identify the identifiers and operators present. Print them separately.




kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
Open
arith_expr.l
~/Desktop/Test
Save
11 }
12
13 %}
14
15 %%
16 [a-zA-Z_][a-zA-Z0-9_]* { printIdentifier(yytext); }
17 "+" { printOperator('+'); }
18 "-" { printOperator('-'); }
19 "*" { printOperator('*'); }
20 "/" { printOperator('/'); }
21 "=" { printOperator('='); }
22 "(" { printOperator('('); }
23 ")" { printOperator(')'); }
24 [\t\n] { /* Ignore whitespace */ }
25 [0-9]+ { /* Ignore numbers */ }
26 . { printf("Invalid character: %s\n", yytext); }
27 %%
28
29 int yywrap(void) {
30 return 1;
31 }
32
33 int main(int argc, char *argv[]) {
34 if (argc > 1) {
35 FILE *file = fopen(argv[1], "r");
36 if (!file) {
37 fprintf(stderr, "Error opening file: %s\n", argv[1]);
38 return 1;
39 }
40 yyin = file;
41 yylex();
42 fclose(file);
43 } else {
44 yylex(); /* read from stdin */
45 }
46 return 0;
47 }
Lex
Tab Width: 8
"arith_expr.l" | 1.1 KiB (1,141 bytes) | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

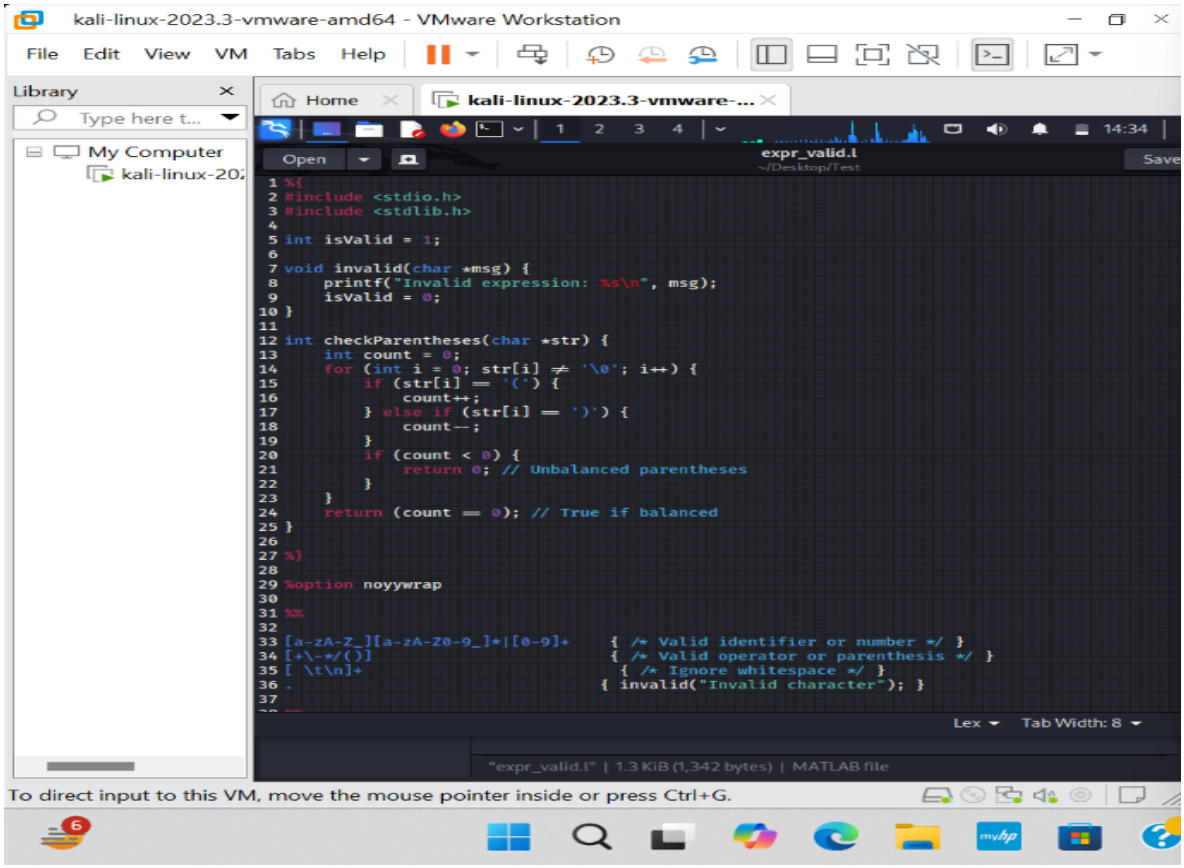

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
1 2 3 4
14:25
Open
*expression.txt
~/Desktop/Test
Log Out...
1 x = (y * z) + 10 / w - temp
2 count = value1 + value2
3 average = (sum1 + sum2) / 2
4 result = a * (b - c)
Plain Text
Tab Width: 8
"expression.txt" | 0 bytes | empty document
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

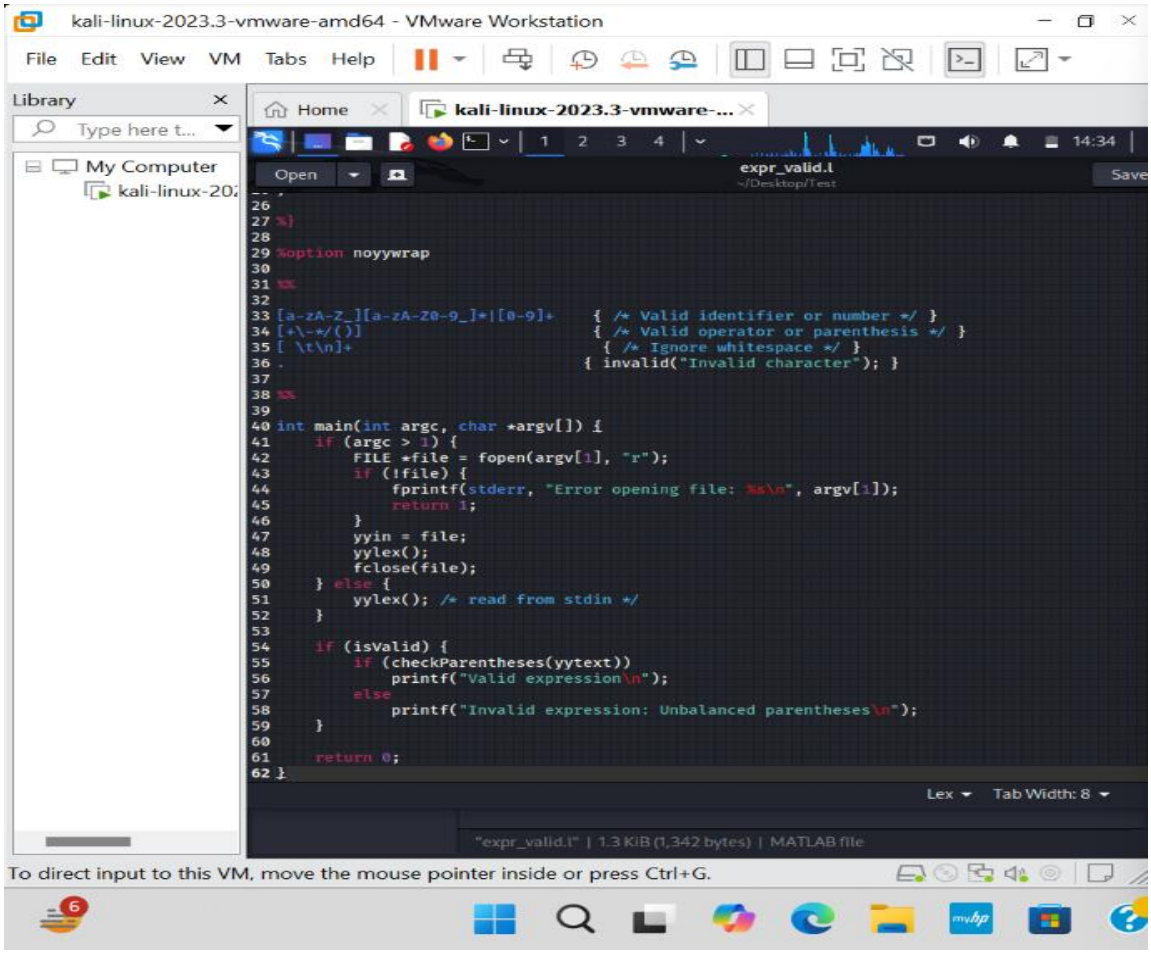

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex arith_expr.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o arith_expr -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./arith_expr expression.txt
Identifier: x
Operator: =
Operator: (
Identifier: y
Operator: *
Identifier: z
Operator:)
Operator: +
Operator: /
Identifier: w
Operator: -
Identifier: temp
Identifier: count
Operator: =
Identifier: value1
Operator: +
Identifier: value2
Identifier: average
Operator: =
Operator: (
Identifier: sum1
Operator: +
Identifier: sum2
Operator:)
Operator: /
Identifier: result
Operator: =
Identifier: a
Operator: *
Operator: (
Identifier: b
Operator: -
Identifier: c
Operator:)
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

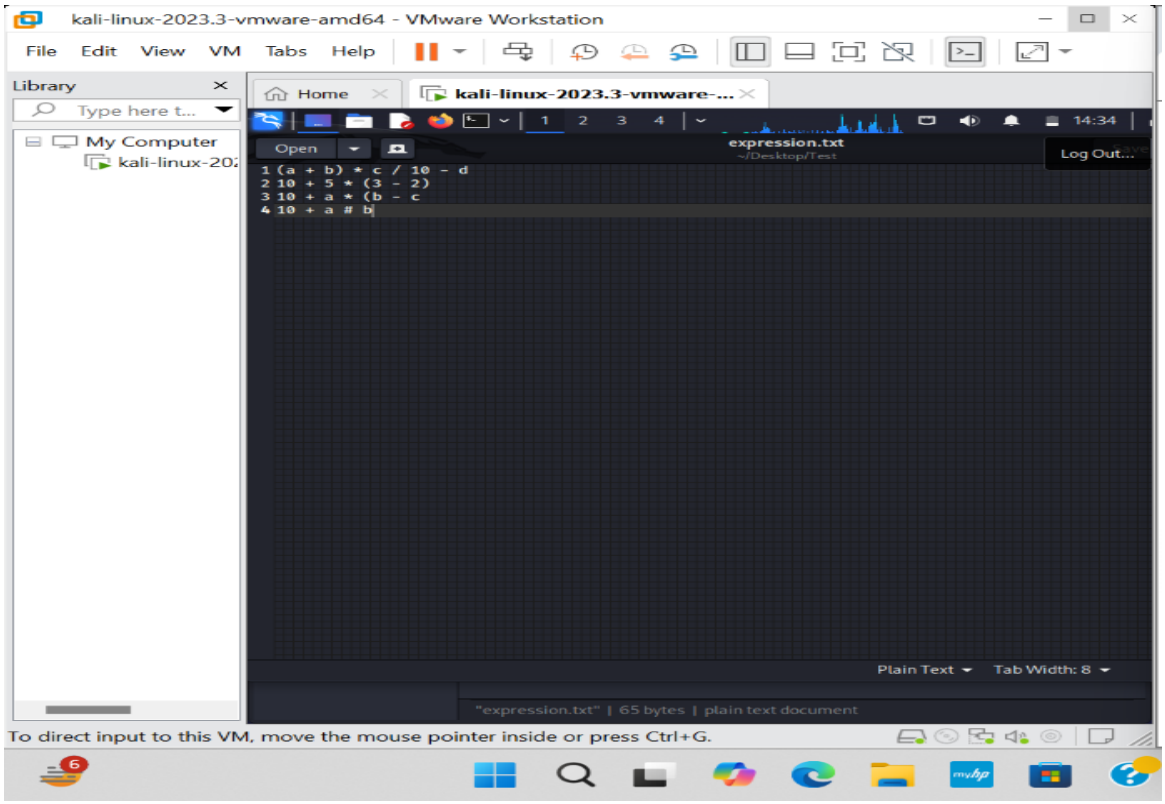
EXPERIMENT NO-4

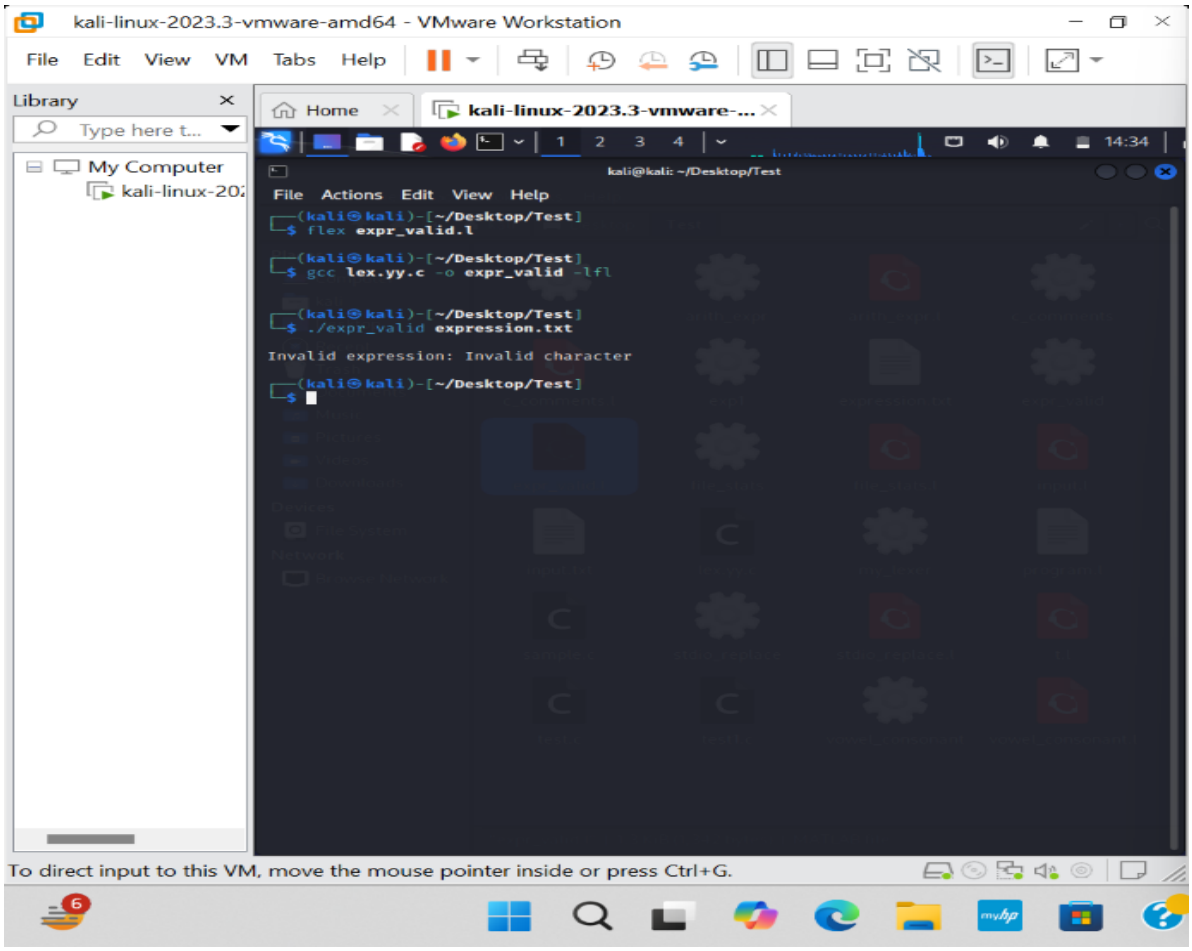
Title: Syntax Analysis via Yacc Tool.

1. WAP to test the validity of a simple expression involving operators- +, -, *, and /.

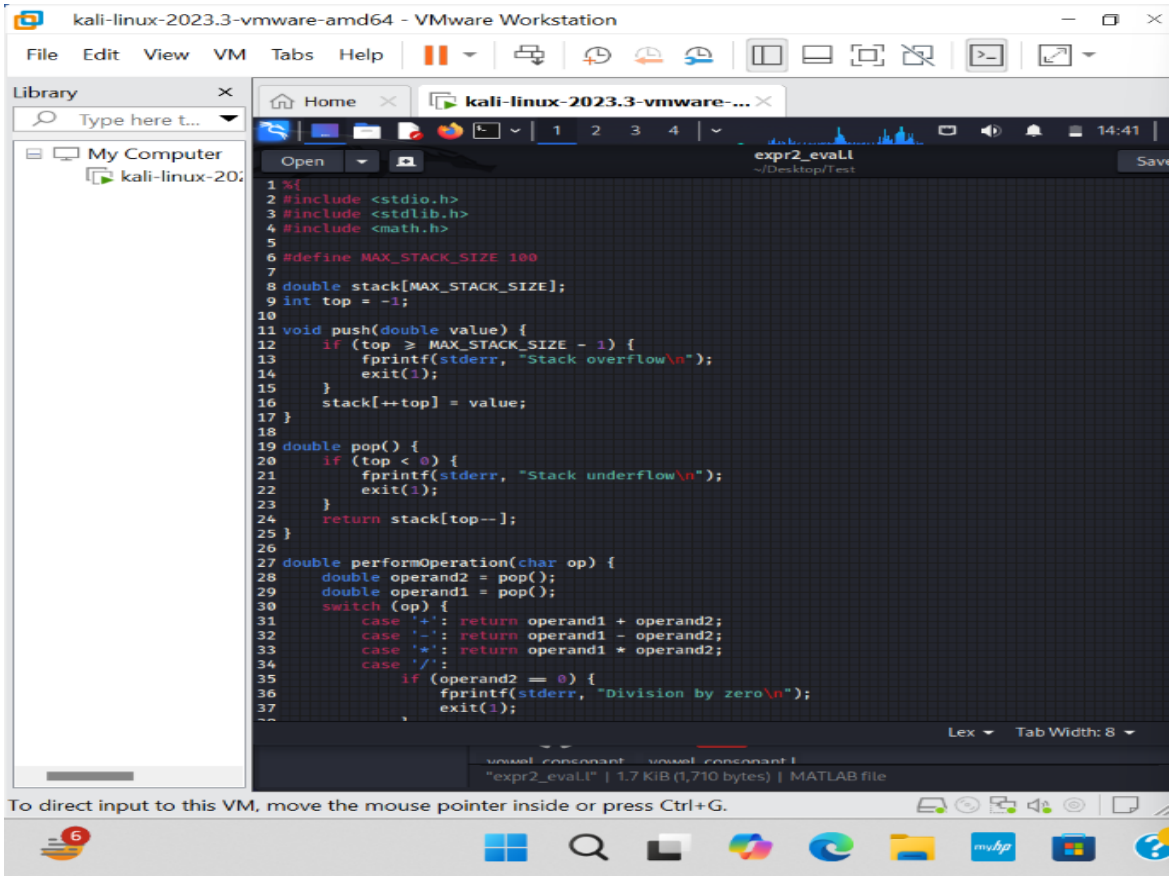


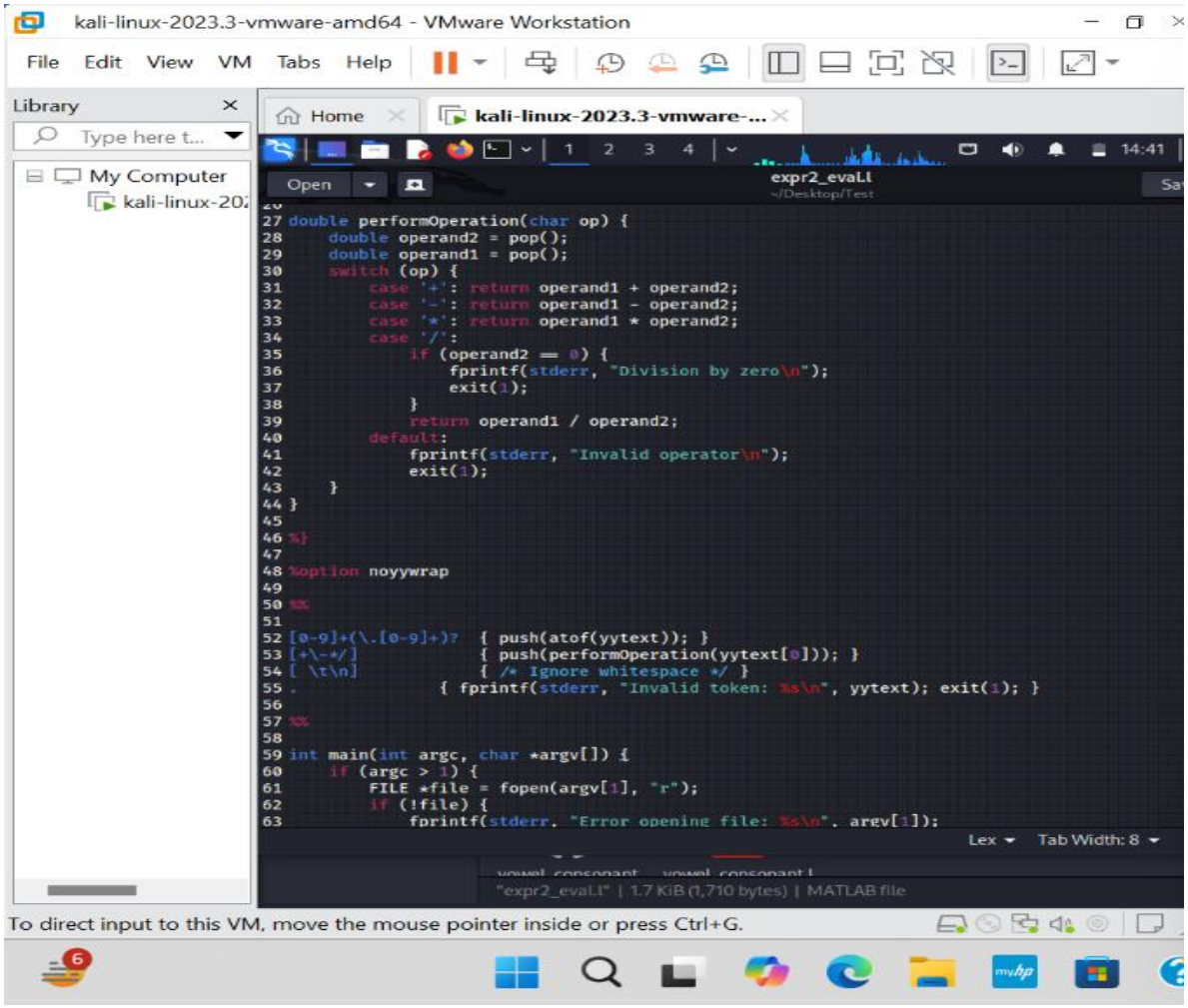

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:34
Open
expr_valid.l
~/Desktop/Test
Save
26
27 %}
28
29 %option noyywrap
30
31 %%
32
33 [a-zA-Z_][a-zA-Z0-9_]*|[0-9]+ { /* Valid identifier or number */ }
34 [+\-*/()] { /* Valid operator or parenthesis */ }
35 [\t\n]+ { /* Ignore whitespace */ }
36 . { invalid("Invalid character"); }
37
38 %%
39
40 int main(int argc, char *argv[]) {
41 if (argc > 1) {
42 FILE *file = fopen(argv[1], "r");
43 if (!file) {
44 fprintf(stderr, "Error opening file: %s\n", argv[1]);
45 return 1;
46 }
47 yyin = file;
48 yylex();
49 fclose(file);
50 } else {
51 yylex(); /* read from stdin */
52 }
53
54 if (isValid) {
55 if (checkParentheses(yytext))
56 printf("Valid expression\n");
57 else
58 printf("Invalid expression: Unbalanced parentheses\n");
59 }
60
61 return 0;
62 }
Lex
Tab Width: 8
"expr_valid.l" | 1.3 KiB (1,342 bytes) | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

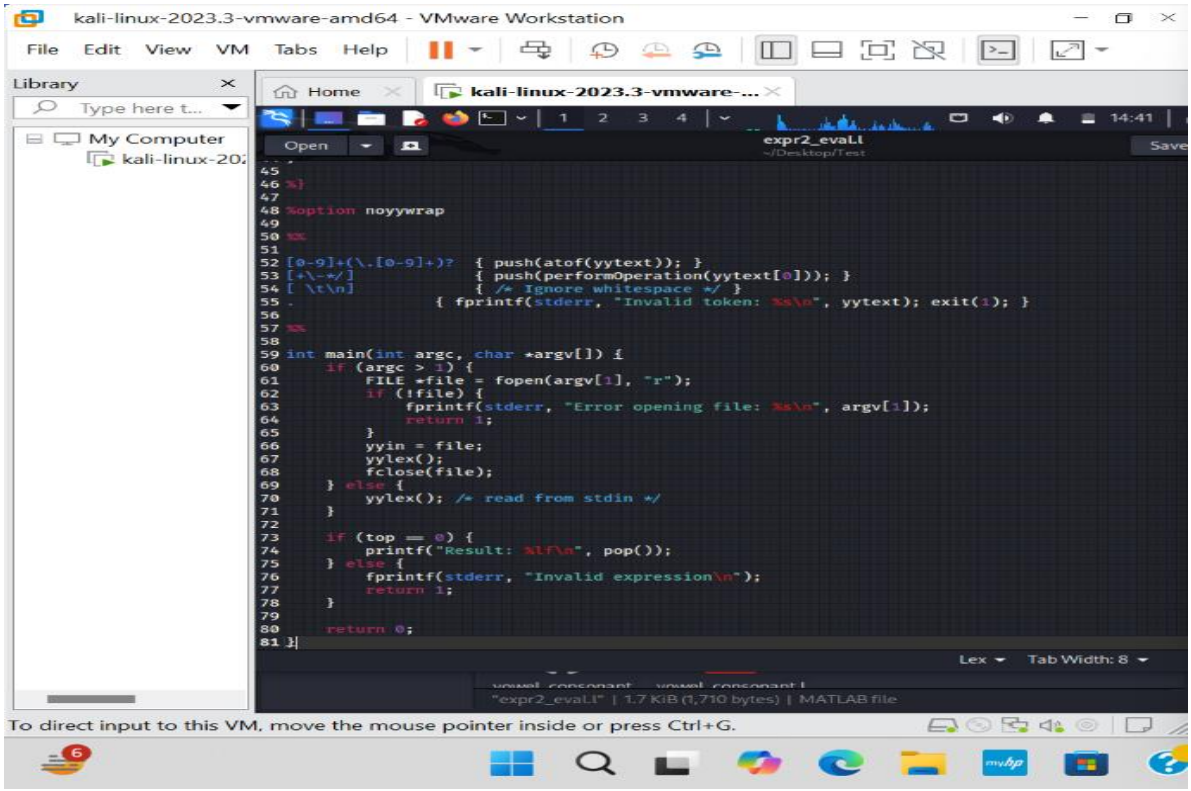

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:34
Open
expression.txt
~/Desktop/Test
Log Out...
1 (a + b) * c / 10 - d
2 10 + 5 * (3 - 2)
3 10 + a * (b - c
4 10 + a # b
Plain Text
Tab Width: 8
"expression.txt" | 65 bytes | plain text document
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

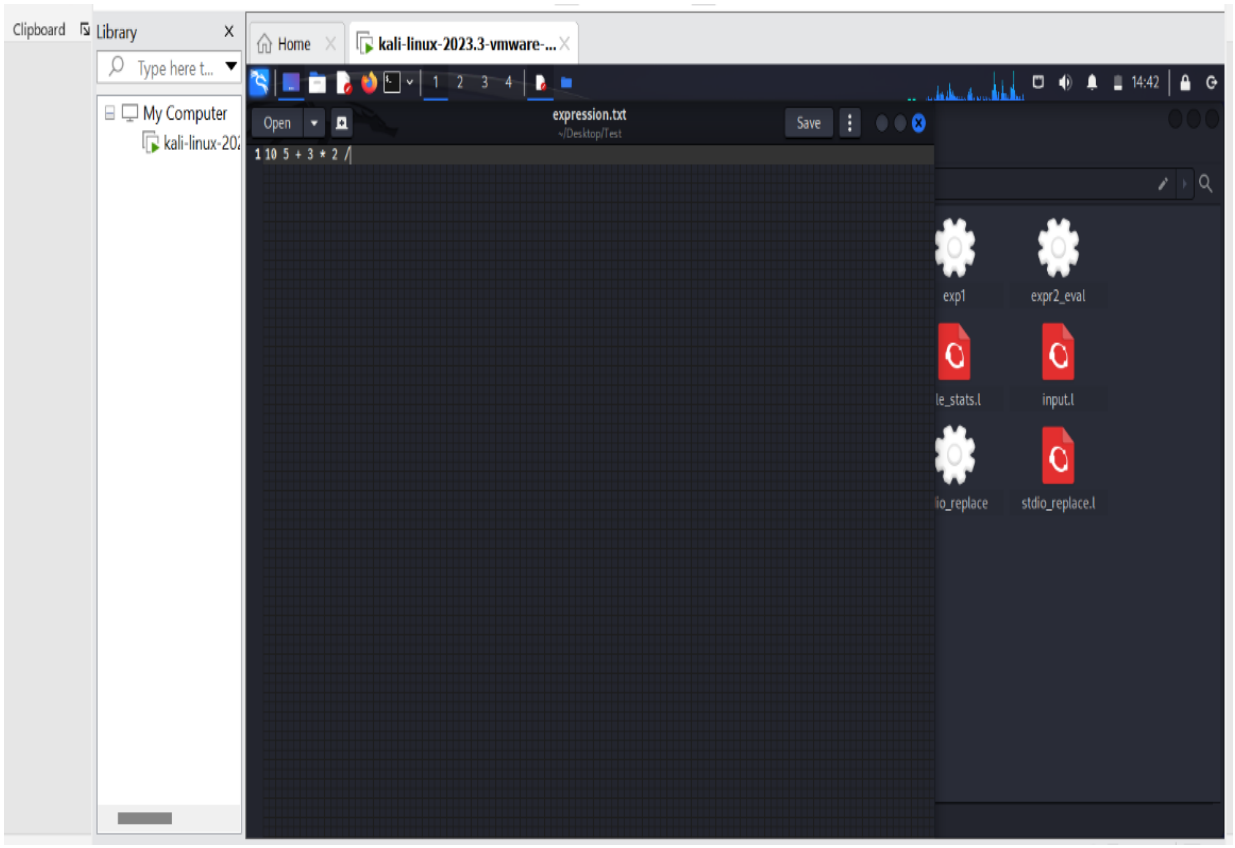

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
14:34
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex expr_valid.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o expr_valid -lfl
(kali㉿kali)-[~/Desktop/Test]
$ ./expr_valid expression.txt
Invalid expression: Invalid character
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

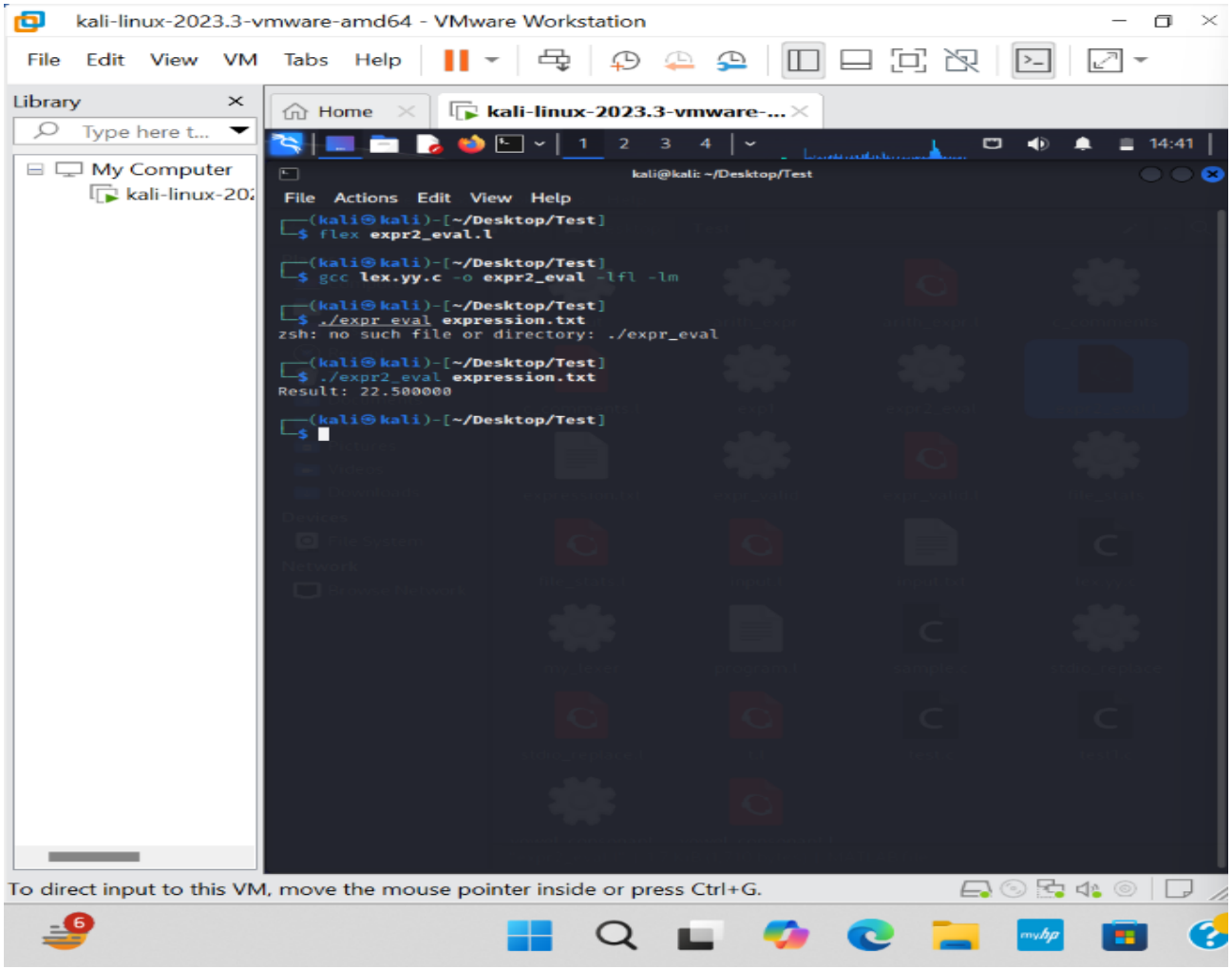
2. WAP to evaluate an arithmetic expression involving operators- +, -, *, and /.




kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
Open
expr2_evaLl
~/Desktop/Test
14:41
double performOperation(char op) {
double operand2 = pop();
double operand1 = pop();
switch (op) {
case '+': return operand1 + operand2;
case '-': return operand1 - operand2;
case '*': return operand1 * operand2;
case '/':
if (operand2 == 0) {
fprintf(stderr, "Division by zero\n");
exit(1);
}
return operand1 / operand2;
default:
fprintf(stderr, "Invalid operator\n");
exit(1);
}
}
%}
%option noyywrap
%%
[0-9]+(\.[0-9]+)? { push(atof(yytext)); }
[+\-*/] { push(performOperation(yytext[0])); }
[\t\n] { /* Ignore whitespace */ }
. { fprintf(stderr, "Invalid token: %s\n", yytext); exit(1); }
%%
int main(int argc, char *argv[]) {
if (argc > 1) {
FILE *file = fopen(argv[1], "r");
if (!file) {
fprintf(stderr, "Error opening file: %s\n", argv[1]);
Lex
Tab Width: 8
"expr2_evaLl" | 1.7 KiB (1,710 bytes) | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
Open
expr2_eval.l
~/Desktop/Test
Save
14:41
%option noyywrap
%%
[0-9]+(\.[0-9]+)? { push(atof(yytext)); }
[+\-*/] { push(performOperation(yytext[0])); }
[\t\n] { /* Ignore whitespace */ }
. { fprintf(stderr, "Invalid token: %s\n", yytext); exit(1); }
%%
int main(int argc, char *argv[]) {
if (argc > 1) {
FILE *file = fopen(argv[1], "r");
if (!file) {
fprintf(stderr, "Error opening file: %s\n", argv[1]);
return 1;
}
yyin = file;
yylex();
fclose(file);
} else {
yylex(); /* read from stdin */
}
if (top == 0) {
printf("Result: %lf\n", pop());
} else {
fprintf(stderr, "Invalid expression\n");
return 1;
}
return 0;
}
Lex
Tab Width: 8
"expr2_eval.l" | 1.7 KiB (1,710 bytes) | MATLAB file
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

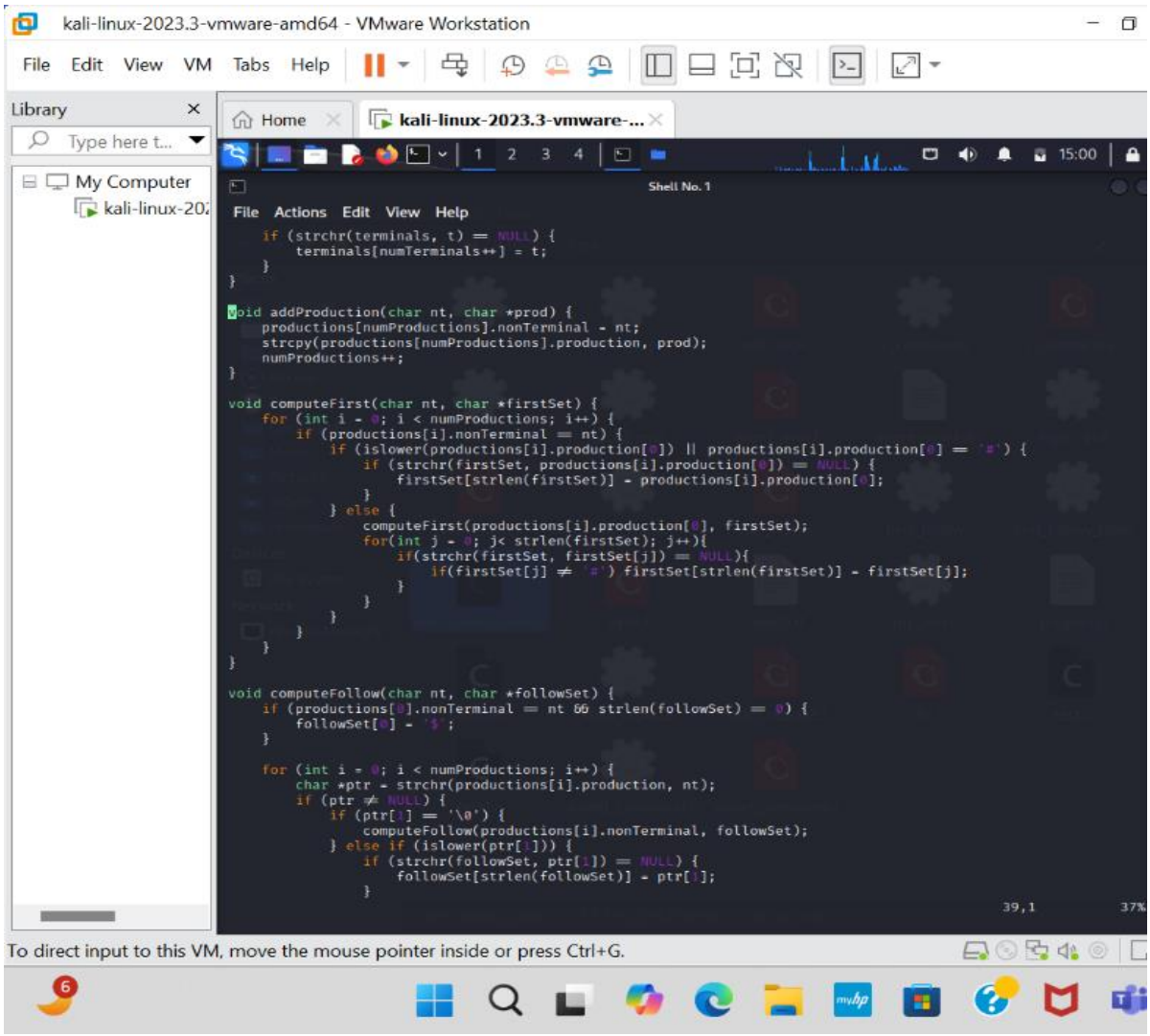

Clipboard
Library
Type here t...
My Computer
kali-linux-202
Home
kali-linux-2023.3-vmware-...
Open
expression.txt
~/Desktop/Test
Save
14:42
1 10 5 + 3 * 2 /
exp1
expr2_eval
le_stats.l
input.l
io_replace
stdio_replace.l

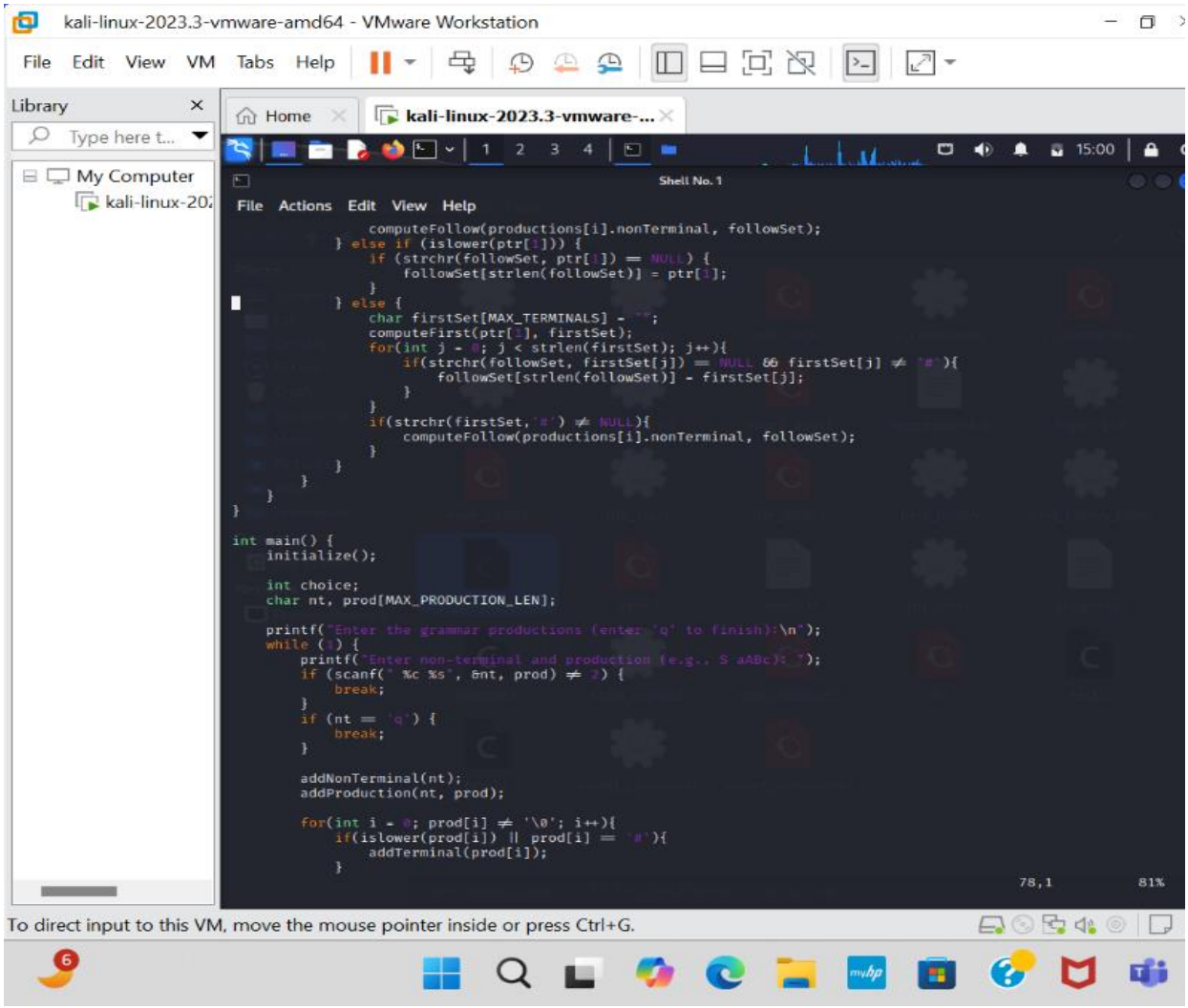

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
14:41
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ flex expr2_eval.l
(kali㉿kali)-[~/Desktop/Test]
$ gcc lex.yy.c -o expr2_eval -lfl -lm
(kali㉿kali)-[~/Desktop/Test]
$ ./expr_eval expression.txt
zsh: no such file or directory: ./expr_eval
(kali㉿kali)-[~/Desktop/Test]
$ ./expr2_eval expression.txt
Result: 22.500000
(kali㉿kali)-[~/Desktop/Test]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

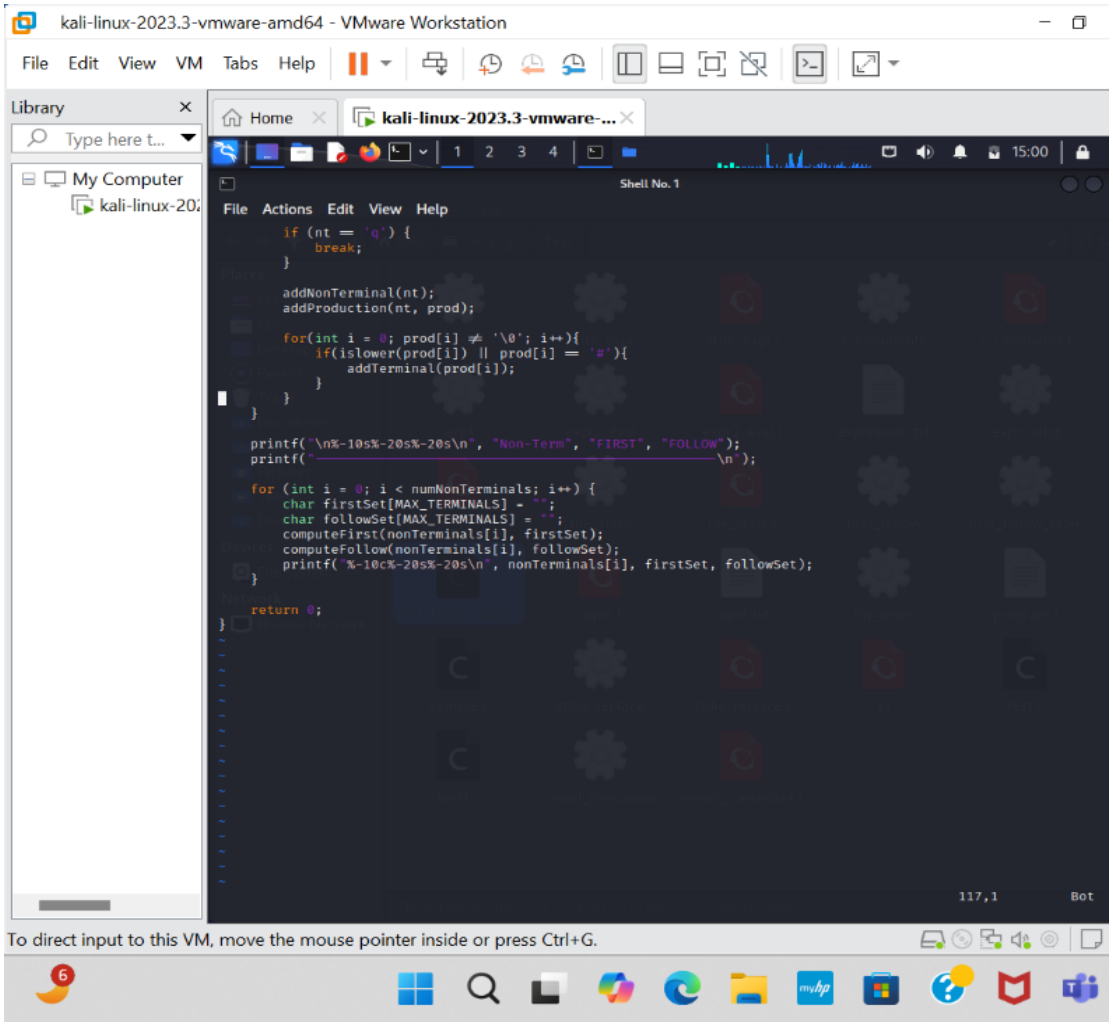
EXPERIMENT NO-5

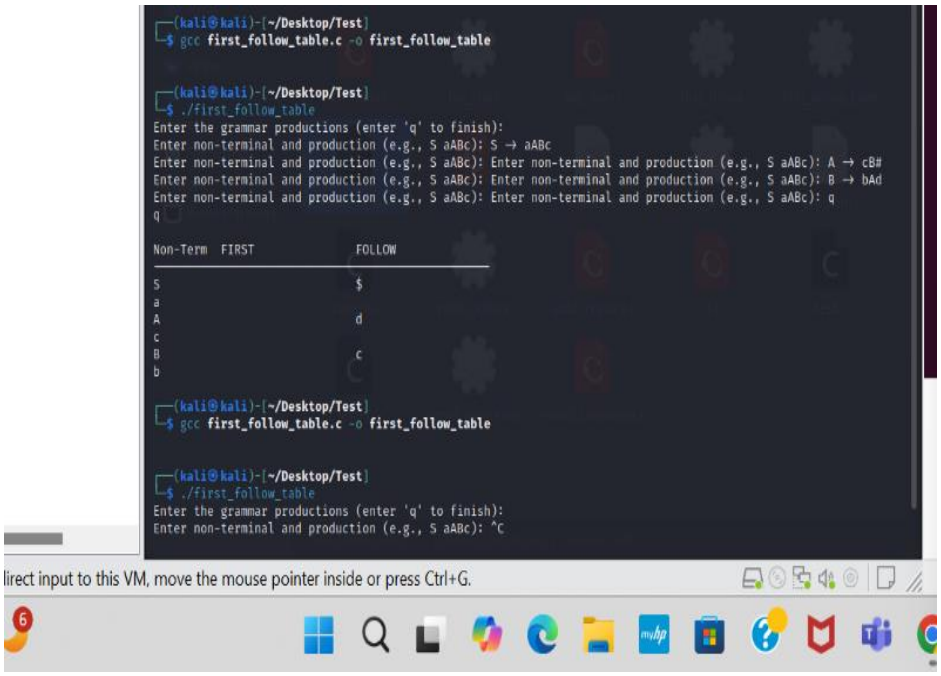
Title: Simulation of FIRST & FOLLOW of a Grammar.

1. WAP to simulate FIRST and FOLLOW of a grammar in C.



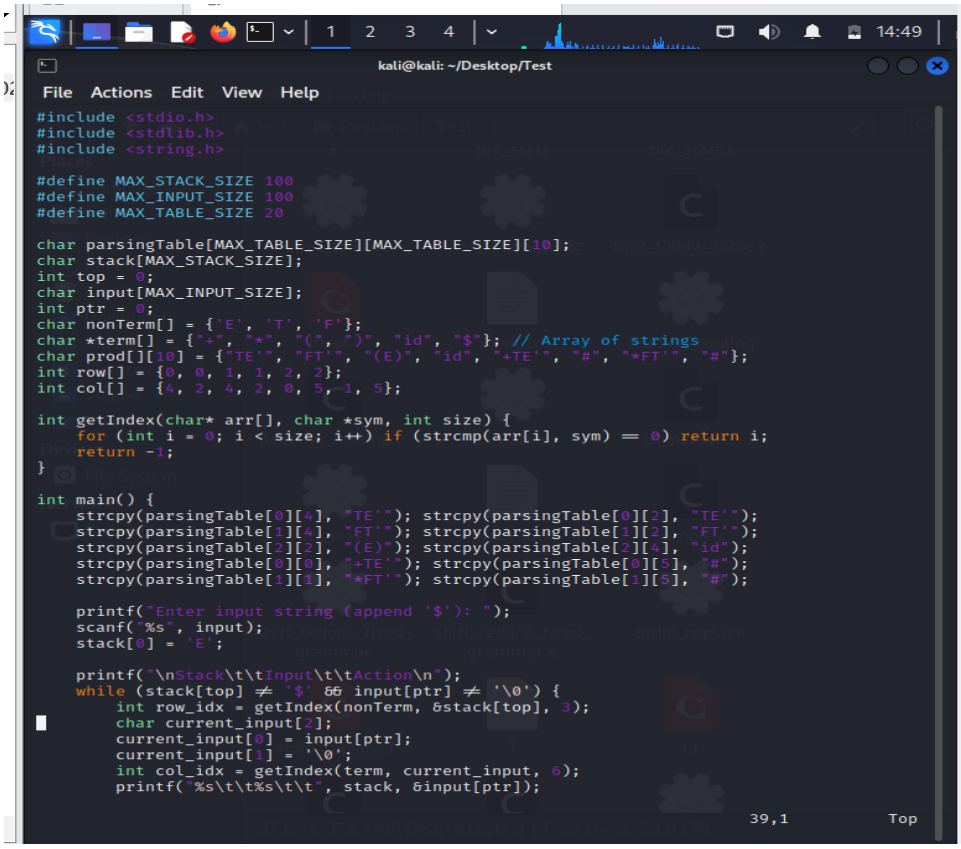

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
Shell No. 1
File Actions Edit View Help
computeFollow(productions[i].nonTerminal, followSet);
} else if (islower(ptr[1])) {
if (strchr(followSet, ptr[1]) == NULL) {
followSet[strlen(followSet)] = ptr[1];
}
} else {
char firstSet[MAX_TERMINALS] = "";
computeFirst(ptr[1], firstSet);
for(int j = 0; j < strlen(firstSet); j++){
if(strchr(followSet, firstSet[j]) == NULL && firstSet[j] != '#'){
followSet[strlen(followSet)] = firstSet[j];
}
}
if(strchr(firstSet,'#') != NULL){
computeFollow(productions[i].nonTerminal, followSet);
}
}
}
}
}
int main() {
initialize();
int choice;
char nt, prod[MAX_PRODUCTION_LEN];
printf("Enter the grammar productions (enter 'q' to finish):\n");
while (1) {
printf("Enter non-terminal and production (e.g., S aABc): ");
if (scanf(" %c %s", &nt, prod) != 2) {
break;
}
if (nt == 'q') {
break;
}
addNonTerminal(nt);
addProduction(nt, prod);
for(int i = 0; prod[i] != '\0'; i++){
if(islower(prod[i]) || prod[i] == '#'){
addTerminal(prod[i]);
}
78,1
81%
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

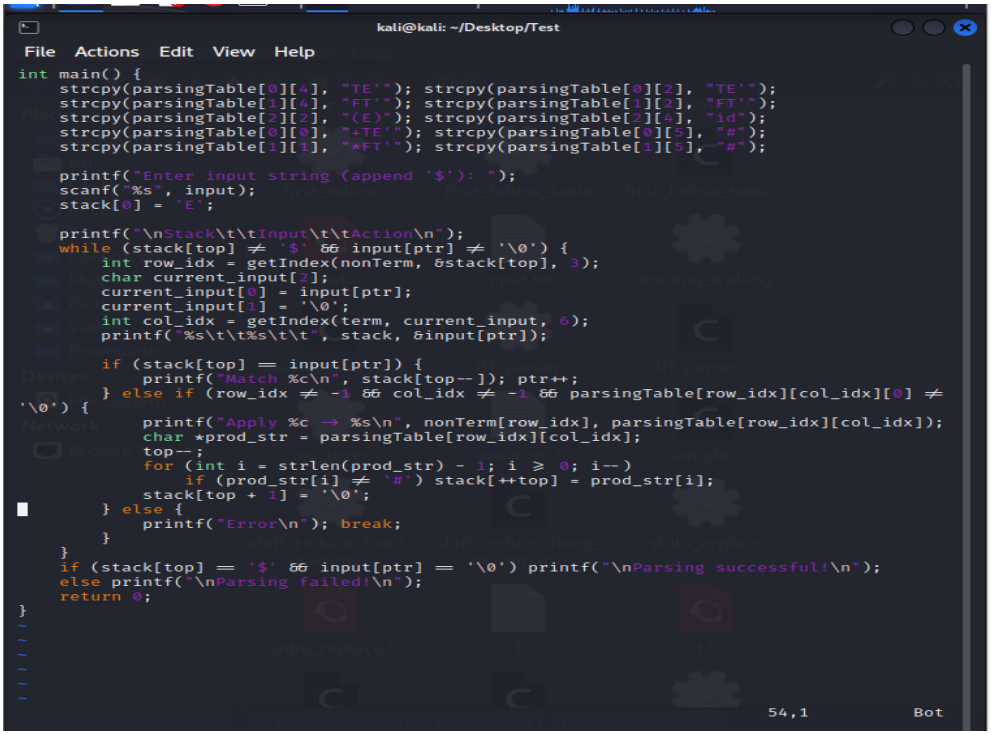

kali-linux-2023.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here t...
My Computer
kali-linux-20
Home
kali-linux-2023.3-vmware-...
Shell No. 1
File Actions Edit View Help
if (nt == 'q') {
break;
}
addNonTerminal(nt);
addProduction(nt, prod);
for(int i = 0; prod[i] != '\0'; i++){
if(islower(prod[i]) || prod[i] == '#'){
addTerminal(prod[i]);
}
}
}
printf("\n%-10s%-20s%-20s\n", "Non-Term", "FIRST", "FOLLOW");
printf("---\n");
for (int i = 0; i < numNonTerminals; i++) {
char firstSet[MAX_TERMINALS] = "";
char followSet[MAX_TERMINALS] = "";
computeFirst(nonTerminals[i], firstSet);
computeFollow(nonTerminals[i], followSet);
printf("%-10c%-20s%-20s\n", nonTerminals[i], firstSet, followSet);
}
return 0;
}
117,1 Bot
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


(kali㉿kali)-[~/Desktop/Test]
$ gcc first_follow_table.c -o first_follow_table
(kali㉿kali)-[~/Desktop/Test]
$ ./first_follow_table
Enter the grammar productions (enter 'q' to finish):
Enter non-terminal and production (e.g., S aABc): S → aABc
Enter non-terminal and production (e.g., S aABc): Enter non-terminal and production (e.g., S aABc): A → cB#
Enter non-terminal and production (e.g., S aABc): Enter non-terminal and production (e.g., S aABc): B → bAd
Enter non-terminal and production (e.g., S aABc): Enter non-terminal and production (e.g., S aABc): q
q
Non-Term FIRST FOLLOW
S $
a
A d
c
B c
b
(kali㉿kali)-[~/Desktop/Test]
$ gcc first_follow_table.c -o first_follow_table
(kali㉿kali)-[~/Desktop/Test]
$ ./first_follow_table
Enter the grammar productions (enter 'q' to finish):
Enter non-terminal and production (e.g., S aABc): ^C
irect input to this VM, move the mouse pointer inside or press Ctrl+G.

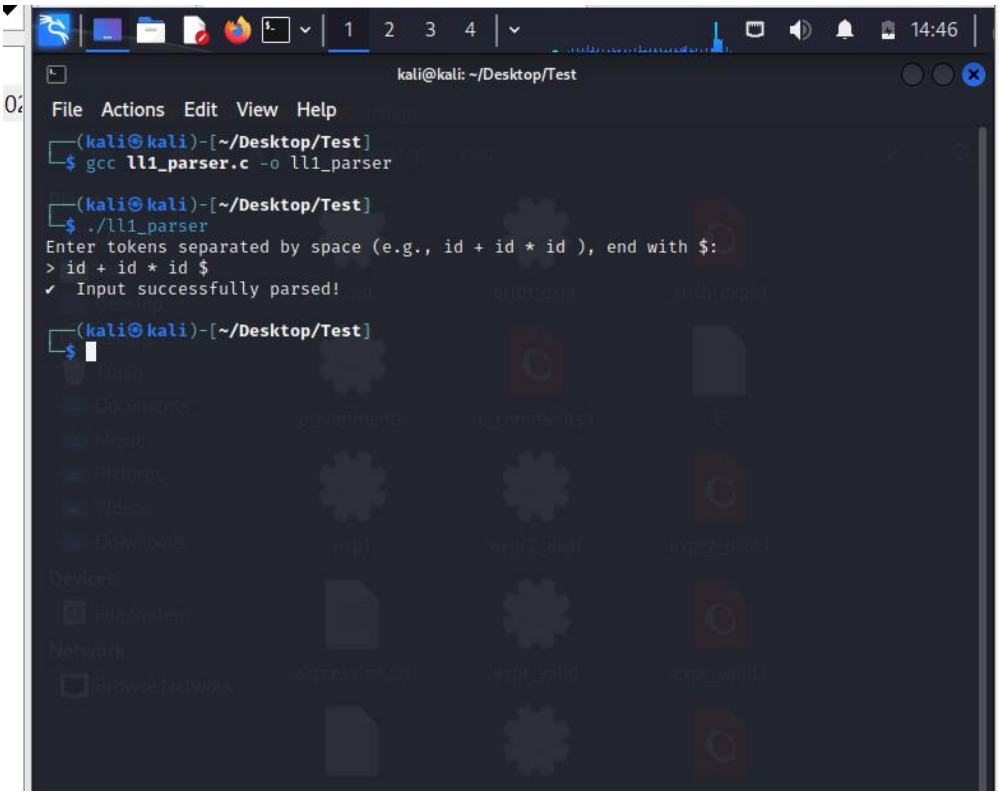
EXPERIMENT-6

1. WAP to construct LL(1) parser for an expression in C



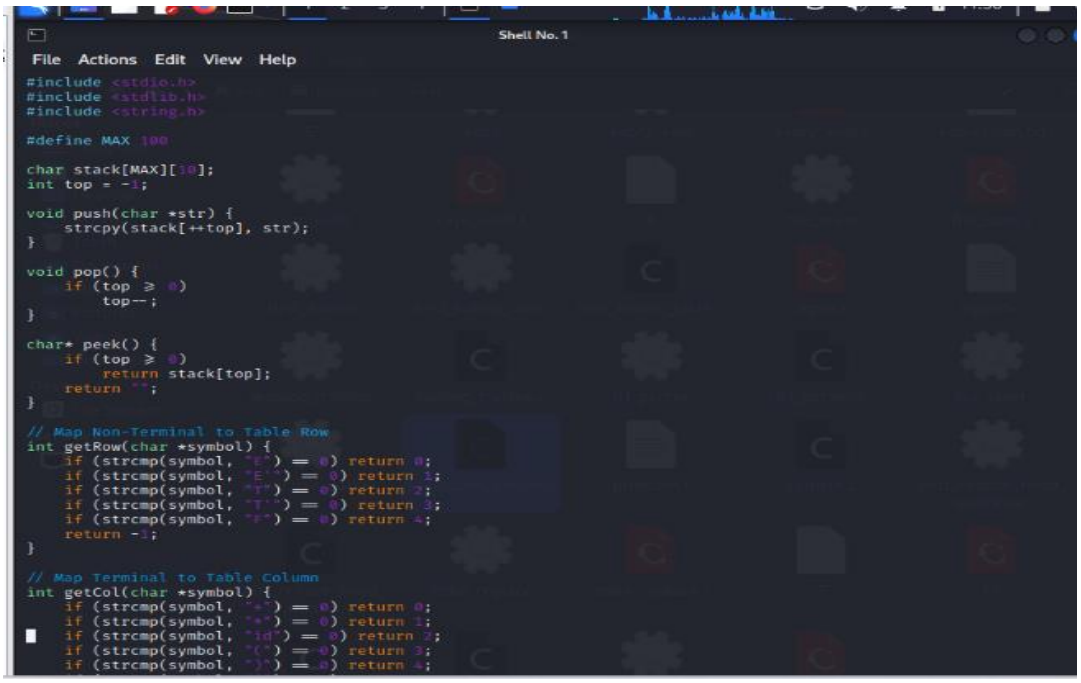


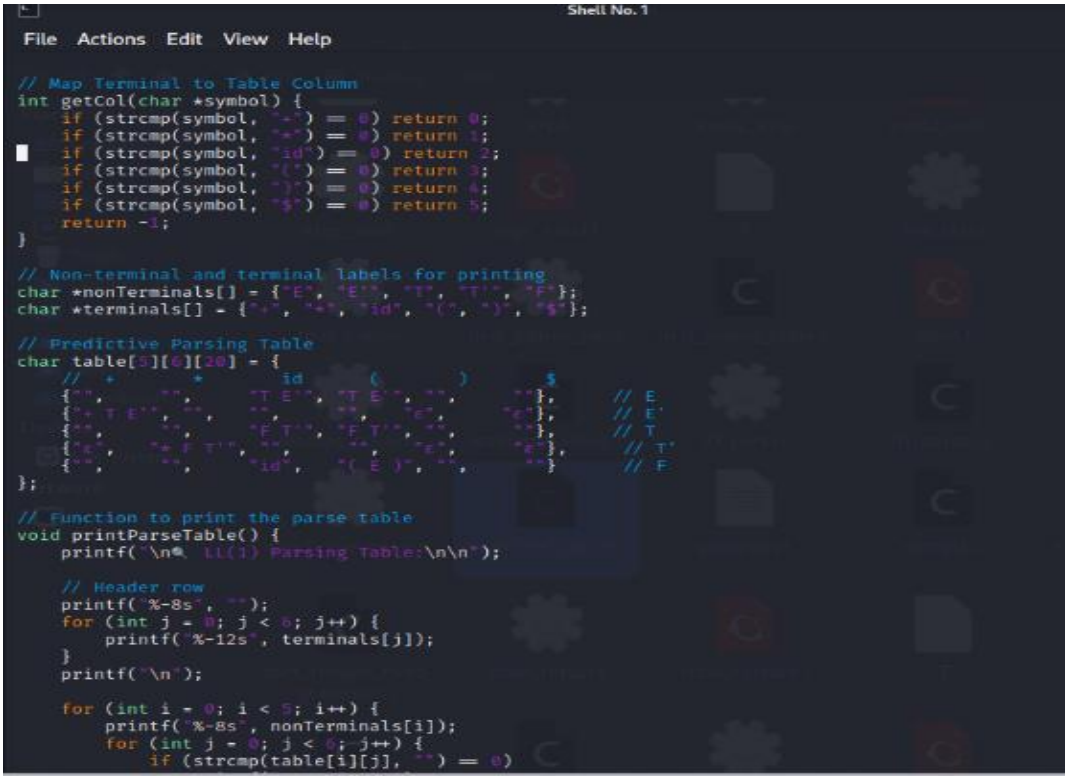

```
int main() {
    strcpy(parsingTable[0][4], "TE'"); strcpy(parsingTable[0][2], "TE'");
    strcpy(parsingTable[1][4], "FT'"); strcpy(parsingTable[1][2], "FT'");
    strcpy(parsingTable[2][2], "(E)"); strcpy(parsingTable[2][4], "id");
    strcpy(parsingTable[0][0], "+TE'"); strcpy(parsingTable[0][5], "#");
    strcpy(parsingTable[1][1], "*FT'"); strcpy(parsingTable[1][5], "#");

    printf("Enter input string (append '$'): ");
    scanf("%s", input);
    stack[0] = 'E';

    printf("\nStack\t\tInput\t\tAction\n");
    while (stack[top] != '$' && input[ptr] != '\0') {
        int row_idx = getIndex(nonTerm, &stack[top], 3);
        char current_input[2];
        current_input[0] = input[ptr];
        current_input[1] = '\0';
        int col_idx = getIndex(term, current_input, 6);
        printf("%s\t\t%s\t\t", stack, &input[ptr]);

        if (stack[top] == input[ptr]) {
            printf("Match %c\n", stack[top--]); ptr++;
        } else if (row_idx != -1 && col_idx != -1 && parsingTable[row_idx][col_idx][0] !=
'\0') {
            printf("Apply %c → %s\n", nonTerm[row_idx], parsingTable[row_idx][col_idx]);
            char *prod_str = parsingTable[row_idx][col_idx];
            top--;
            for (int i = strlen(prod_str) - 1; i >= 0; i--)
                if (prod_str[i] != '#') stack[++top] = prod_str[i];
            stack[top + 1] = '\0';
        } else {
            printf("Error\n"); break;
        }
    }
    if (stack[top] == '$' && input[ptr] == '\0') printf("\nParsing successful!\n");
    else printf("\nParsing failed!\n");
    return 0;
}
```



EXPERIMENT-7

Title: Design of a Predictive Parser for a given Language Lab Activity: WAP to implement a Predictive Parser for a given language in C



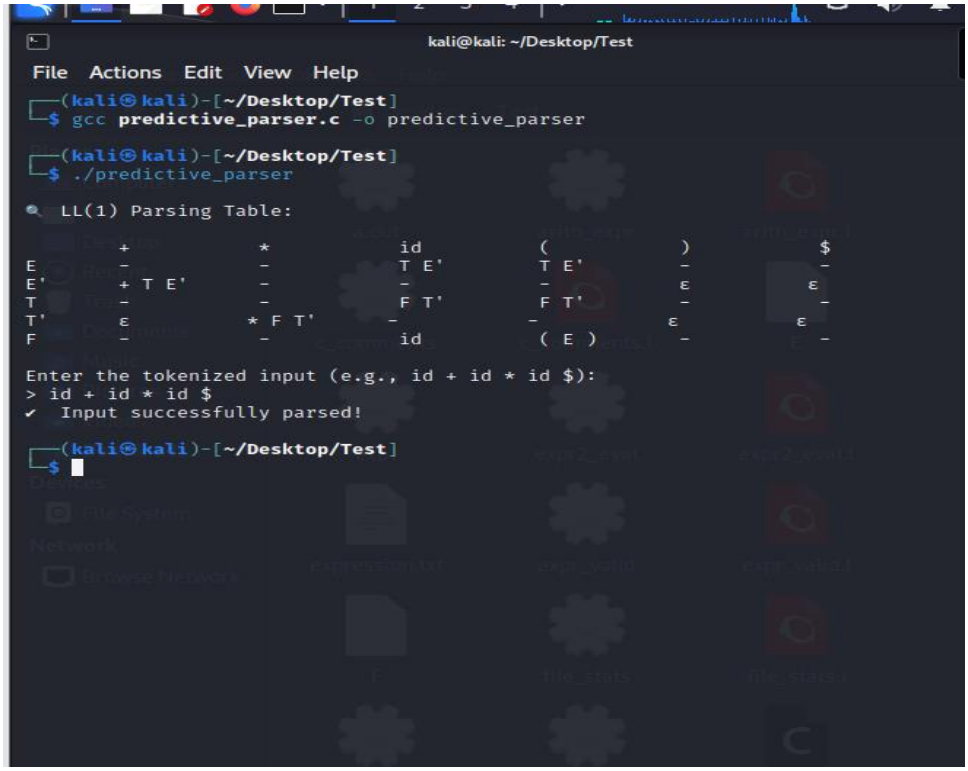


File Actions Edit View Help

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

int main() {
    char input[MAX][10];
    int n = 0;

    // Print the parsing table
    printParseTable();

    printf("Enter the tokenized input (e.g., id + id * id $):\n> ");
    while (1) {
        scanf("%s", input[n]);
        if (strcmp(input[n], "$") == 0)
            break;
        n++;
    }
    strcpy(input[++n], "$");

    push("$");
    push("E");

    int ip = 0;

    while (1) {
        char *X = peek();       // Stack top
        char *a = input[ip];    // Current input symbol

        if (
            strcmp(X, "+") == 0 || strcmp(X, "*") == 0 ||
            strcmp(X, "id") == 0 || strcmp(X, "(") == 0 ||
            strcmp(X, ")") == 0 || strcmp(X, "$") == 0
        ) {
            if (strcmp(X, a) == 0) {
                pop();
                ip++;
                if (strcmp(X, "$") == 0) {
                    printf("✓  Input successfully parsed!\n");
                    break;
```

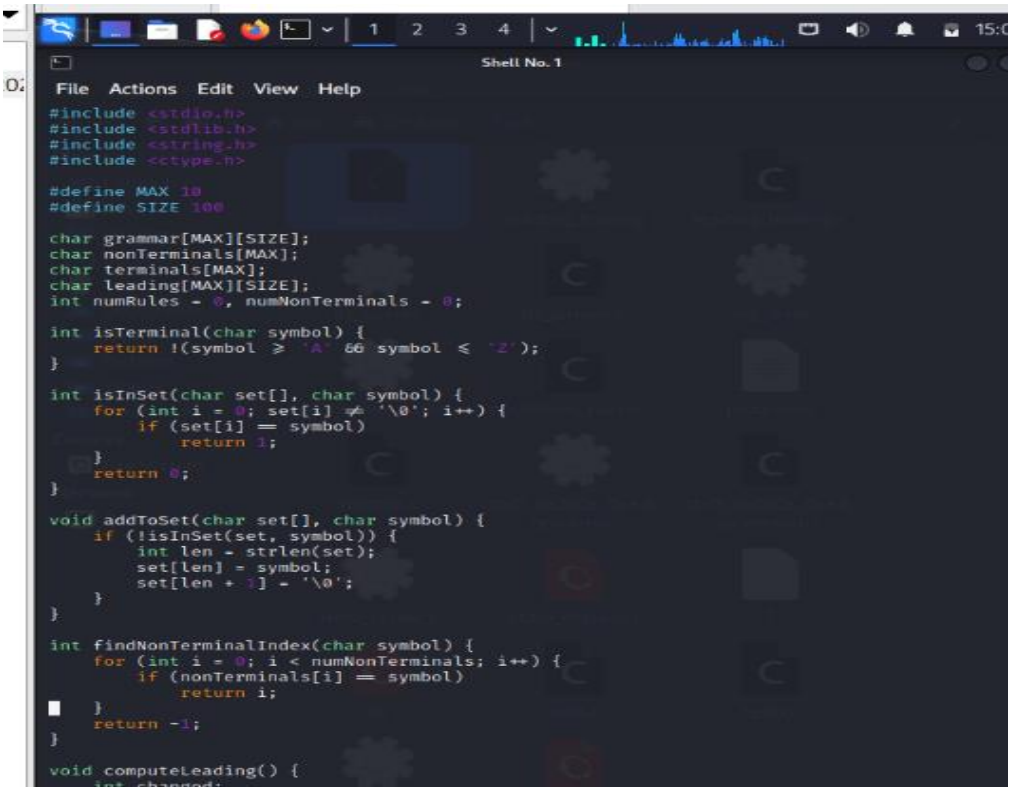
File Actions Edit View Help

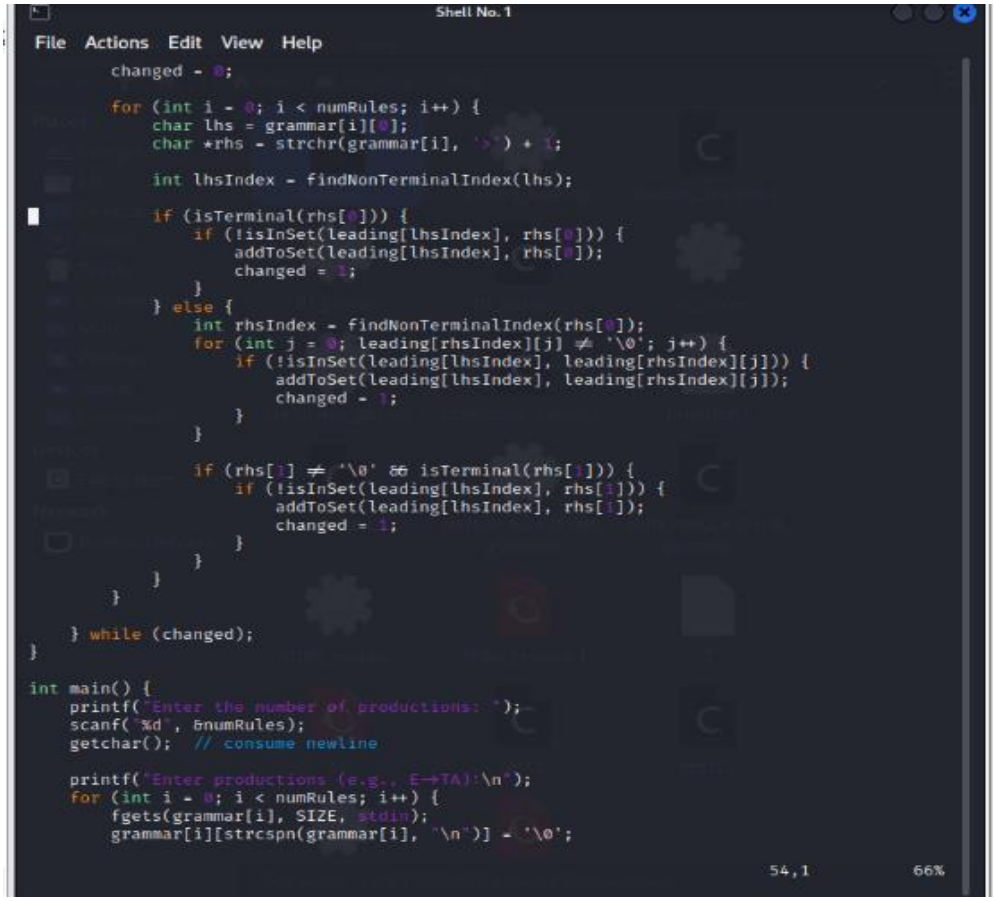
```
                    printf("✓  Input successfully parsed!\n");
                    break;
                }
            } else {
                printf("✗  Error: Mismatched symbol. Stack top: %s, Input: %s\n", X, a);
                break;
            }
        } else {
            int row = getRow(X);
            int col = getCol(a);

            if (row == -1 || col == -1 || strcmp(table[row][col], "") == 0) {
                printf("✗  Error: No rule for [%s, %s]\n", X, a);
                break;
            }

            pop();

            if (strcmp(table[row][col], "ε") != 0) {
                char production[20];
                strcpy(production, table[row][col]);

                char *symbols[5];
                int count = 0;
                char *token = strtok(production, " ");
                while (token != NULL) {
                    symbols[count++] = token;
                    token = strtok(NULL, " ");
                }

                for (int i = count - 1; i >= 0; i--) {
                    push(symbols[i]);
                }
            }
        }
    }

    return 0;
}
```

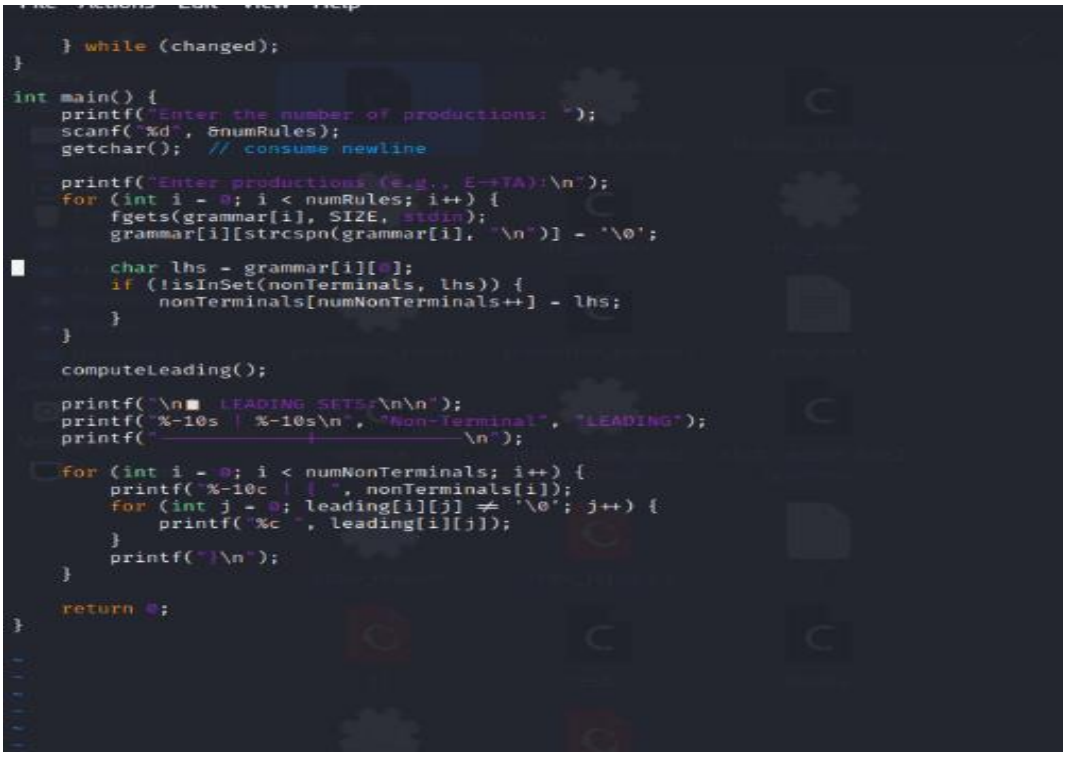

kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ gcc predictive_parser.c -o predictive_parser
(kali㉿kali)-[~/Desktop/Test]
$ ./predictive_parser
LL(1) Parsing Table:
+ * id ( ) $
E - - T E' T E' - -
E' + T E' - - - ε ε
T - - F T' F T' - -
T' ε * F T' - - ε ε
F - - id (E) - -
Enter the tokenized input (e.g., id + id * id $):
> id + id * id $
✔ Input successfully parsed!
(kali㉿kali)-[~/Desktop/Test]
$

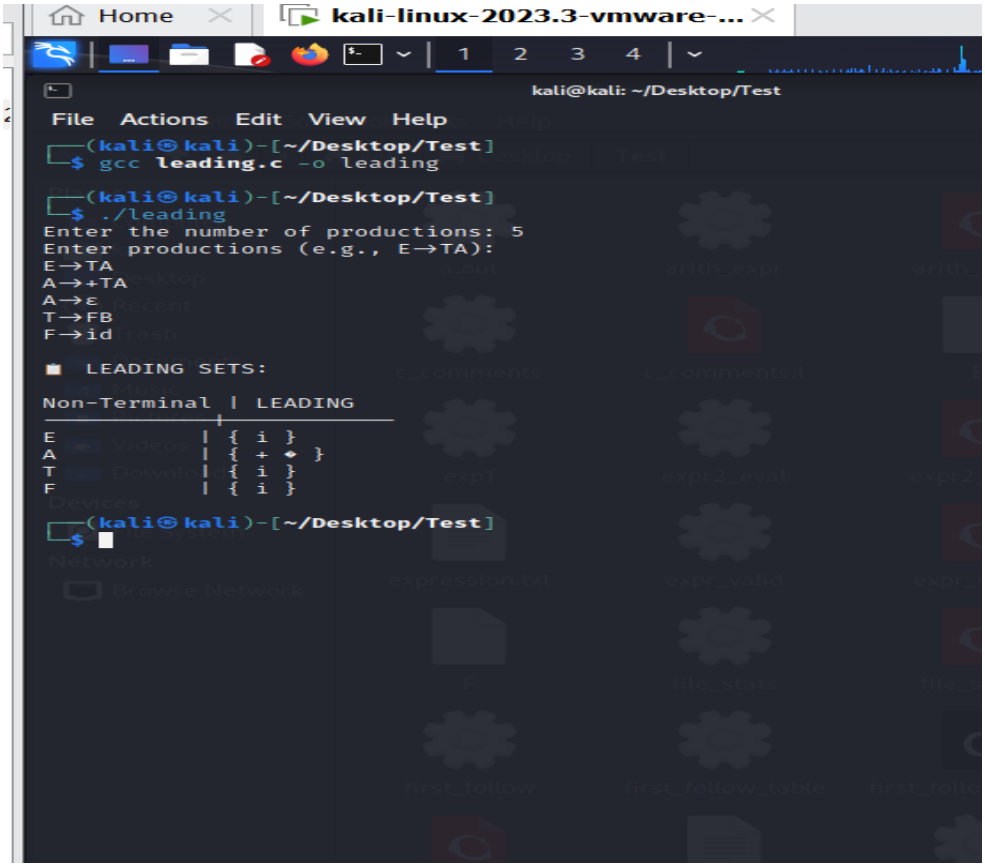
EXPERIMENT-8

Title: Calculation of Leading for all the Non-Terminals Lab Activity: WAP to calculate Leading for all the Non-Terminals in C




Shell No. 1
File Actions Edit View Help
changed = 0;
for (int i = 0; i < numRules; i++) {
char lhs = grammar[i][0];
char *rhs = strchr(grammar[i], '>') + 1;
int lhsIndex = findNonTerminalIndex(lhs);
if (isTerminal(rhs[0])) {
if (!isInSet(leading[lhsIndex], rhs[0])) {
addToSet(leading[lhsIndex], rhs[0]);
changed = 1;
}
} else {
int rhsIndex = findNonTerminalIndex(rhs[0]);
for (int j = 0; leading[rhsIndex][j] != '\0'; j++) {
if (!isInSet(leading[lhsIndex], leading[rhsIndex][j])) {
addToSet(leading[lhsIndex], leading[rhsIndex][j]);
changed = 1;
}
}
if (rhs[1] != '\0' && isTerminal(rhs[1])) {
if (!isInSet(leading[lhsIndex], rhs[1])) {
addToSet(leading[lhsIndex], rhs[1]);
changed = 1;
}
}
}
}
} while (changed);
}
int main() {
printf("Enter the number of productions: ");
scanf("%d", &numRules);
getchar(); // consume newline
printf("Enter productions (e.g., E->TA):\n");
for (int i = 0; i < numRules; i++) {
fgets(grammar[i], SIZE, stdin);
grammar[i][strcspn(grammar[i], "\n")] = '\0';
54,1 66%

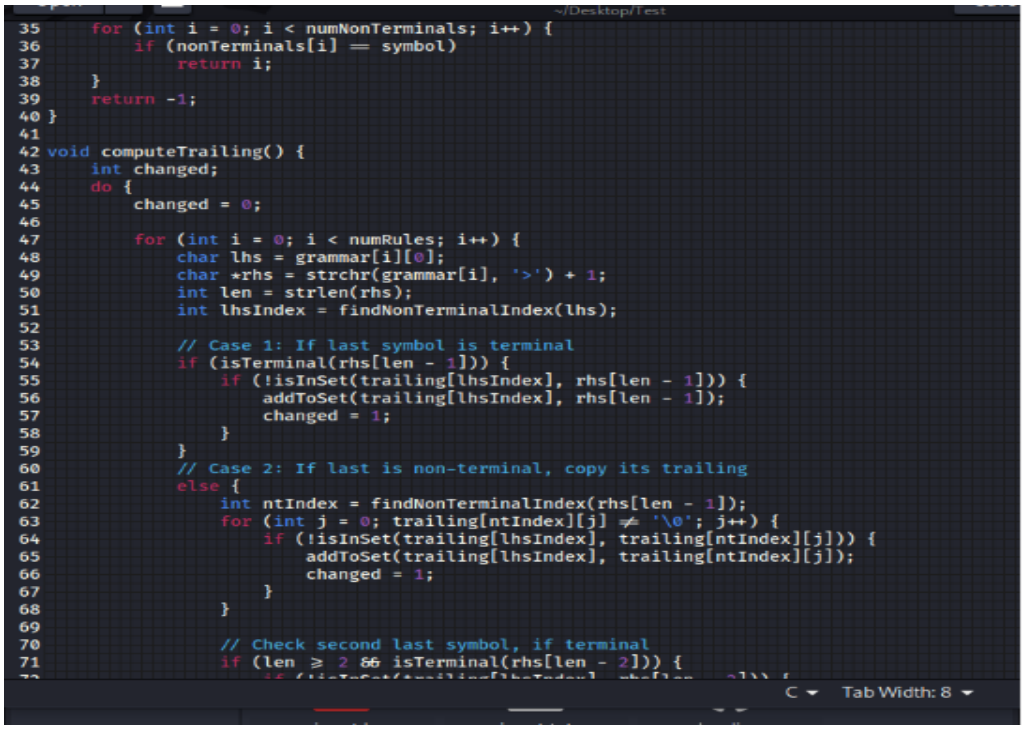

} while (changed);
}
int main() {
printf("Enter the number of productions: ");
scanf("%d", &numRules);
getchar(); // consume newline
printf("Enter productions (e.g., E->TA):\n");
for (int i = 0; i < numRules; i++) {
fgets(grammar[i], SIZE, stdin);
grammar[i][strcspn(grammar[i], "\n")] = '\0';
char lhs = grammar[i][0];
if (!isInSet(nonTerminals, lhs)) {
nonTerminals[numNonTerminals++] = lhs;
}
}
computeLeading();
printf("\n LEADING SETS:\n\n");
printf("%-10s | %-10s\n", "Non-Terminal", "LEADING");
printf("-----------+-----------\n");
for (int i = 0; i < numNonTerminals; i++) {
printf("%-10c | { ", nonTerminals[i]);
for (int j = 0; leading[i][j] != '\0'; j++) {
printf("%c ", leading[i][j]);
}
printf("}\n");
}
return 0;
}


Home
kali-linux-2023.3-vmware-...
1 2 3 4
kali@kali: ~/Desktop/Test
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ gcc leading.c -o leading
(kali㉿kali)-[~/Desktop/Test]
$ ./leading
Enter the number of productions: 5
Enter productions (e.g., E→TA):
E→TA
A→+TA
A→ε
T→FB
F→id
LEADING SETS:
Non-Terminal | LEADING
E | { i }
A | { + ε }
T | { i }
F | { i }
(kali㉿kali)-[~/Desktop/Test]
$

EXPERIMENT-9

Title: Calculation of Trailing for all the Non-Terminals Lab Activity: WAP to calculate Trailing for all the Non Terminals in C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX 10
#define SIZE 100

char grammar[MAX][SIZE];
char nonTerminals[MAX];
char trailing[MAX][SIZE];
int numRules = 0, numNonTerminals = 0;

int isTerminal(char symbol) {
    return !(symbol >= 'A' && symbol <= 'Z');
}

int isInSet(char set[], char symbol) {
    for (int i = 0; set[i] != '\0'; i++) {
        if (set[i] == symbol)
            return 1;
    }
    return 0;
}

void addToSet(char set[], char symbol) {
    if (!isInSet(set, symbol)) {
        int len = strlen(set);
        set[len] = symbol;
        set[len + 1] = '\0';
    }
}

int findNonTerminalIndex(char symbol) {
    for (int i = 0; i < numNonTerminals; i++) {
        if (nonTerminals[i] == symbol)
            return i;
```



```c
    for (int i = 0; i < numNonTerminals; i++) {
        if (nonTerminals[i] == symbol)
            return i;
    }
    return -1;
}

void computeTrailing() {
    int changed;
    do {
        changed = 0;

        for (int i = 0; i < numRules; i++) {
            char lhs = grammar[i][0];
            char *rhs = strchr(grammar[i], '>') + 1;
            int len = strlen(rhs);
            int lhsIndex = findNonTerminalIndex(lhs);

            // Case 1: If last symbol is terminal
            if (isTerminal(rhs[len - 1])) {
                if (!isInSet(trailing[lhsIndex], rhs[len - 1])) {
                    addToSet(trailing[lhsIndex], rhs[len - 1]);
                    changed = 1;
                }
            }
            // Case 2: If last is non-terminal, copy its trailing
            else {
                int ntIndex = findNonTerminalIndex(rhs[len - 1]);
                for (int j = 0; trailing[ntIndex][j] != '\0'; j++) {
                    if (!isInSet(trailing[lhsIndex], trailing[ntIndex][j])) {
                        addToSet(trailing[lhsIndex], trailing[ntIndex][j]);
                        changed = 1;
                    }
                }

                // Check second last symbol, if terminal
                if (len >= 2 && isTerminal(rhs[len - 2])) {
```

```c

    } while (changed);
}

int main() {
    printf("Enter the number of productions: ");
    scanf("%d", &numRules);
    getchar(); // consume newline

    printf("Enter productions (e.g., E→TA):\n");
    for (int i = 0; i < numRules; i++) {
        fgets(grammar[i], SIZE, stdin);
        grammar[i][strcspn(grammar[i], "\n")] = '\0';

        char lhs = grammar[i][0];
        if (!isInSet(nonTerminals, lhs)) {
            nonTerminals[numNonTerminals++] = lhs;
        }
    }

    computeTrailing();

    printf("\n TRAILING SETS:\n\n");
    printf("%-10s | %-10s\n", "Non-Terminal", "TRAILING");
    printf("-----------+-----------\n");

    for (int i = 0; i < numNonTerminals; i++) {
        printf("%-10c | { ", nonTerminals[i]);
        for (int j = 0; trailing[i][j] != '\0'; j++) {
            printf("%c ", trailing[i][j]);
        }
        printf("}\n");
    }

    return 0;
}
```

er

-202

kali@kali: ~/Desktop/Test

File Actions Edit View Help

```
┌──(kali㉿kali)-[~/Desktop/Test]
└─$ gcc trailing.c -o trailing

┌──(kali㉿kali)-[~/Desktop/Test]
└─$ ./trailing
Enter the number of productions: 5
Enter productions (e.g., E→TA):
E→TA
A→+TA
A→ε
T→FB
F→id

 TRAILING SETS:

Non-Terminal | TRAILING
-------------+-----------
E            | { $ }
A            | { $ }
T            | { }
F            | { d }

┌──(kali㉿kali)-[~/Desktop/Test]
└─$
```

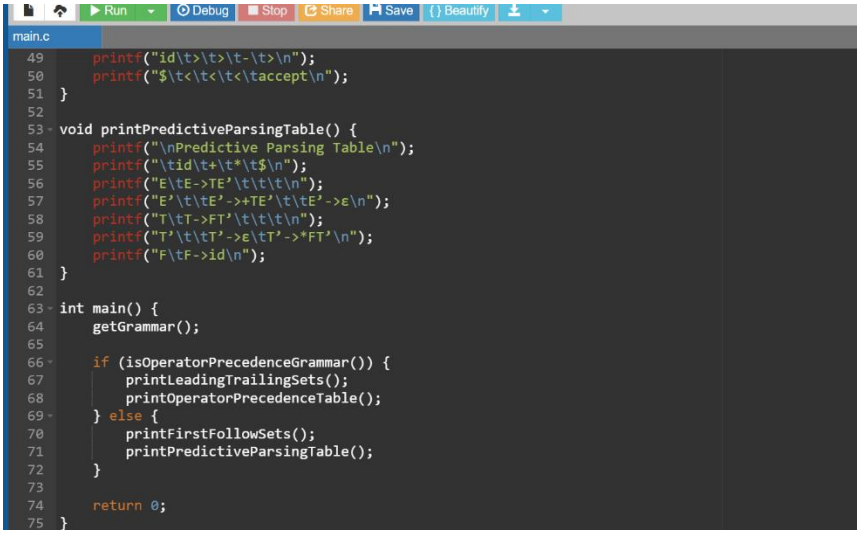
EXPERIMENT-10

Title: Development of an Operator Precedence Parser Lab Activity: WAP to implement an Operator Precedence Parser for a given language in C

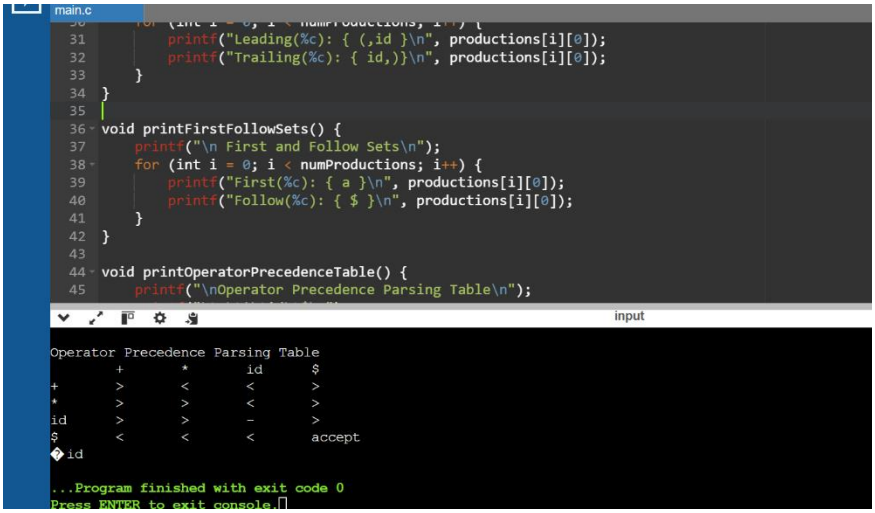
main.c

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX 10

char nonTerminals[MAX][10], terminals[MAX][10], productions[MAX][50];
int numProductions = 0;

void getGrammar() {
    printf("Enter number of productions: ");
    scanf("%d", &numProductions);
    printf("Enter productions:\n");
    for (int i = 0; i < numProductions; i++) {
        scanf("%s", productions[i]);
    }
}

int isOperatorPrecedenceGrammar() {
    for (int i = 0; i < numProductions; i++) {
        if (strchr(productions[i], '|') != NULL) {
            return 0;
        }
    }
    return 1;
}
```

```c
void printLeadingTrailingSets() {
    printf("\n Leading and Trailing Sets\n");
    for (int i = 0; i < numProductions; i++) {
        printf("Leading(%c): { (,id }\n", productions[i][0]);
        printf("Trailing(%c): { id,)}\n", productions[i][0]);
    }
}

void printFirstFollowSets() {
    printf("\n First and Follow Sets\n");
    for (int i = 0; i < numProductions; i++) {
        printf("First(%c): { a }\n", productions[i][0]);
        printf("Follow(%c): { $ }\n", productions[i][0]);
    }
}

void printOperatorPrecedenceTable() {
    printf("\nOperator Precedence Parsing Table\n");
    printf("\t+\t*\tid\t$\n");
    printf("+\t>\t<\t<\t>\n");
    printf("*\t>\t>\t<\t>\n");
    printf("id\t>\t>\t-\t>\n");
    printf("$\t<\t<\t<\taccept\n");
}

void printPredictiveParsingTable() {
    printf("\nPredictive Parsing Table\n");
```



```
Run  Debug  Stop  Share  Save  { } Beautify
main.c
49      printf("id\t>\t>\t-\t>\n");
50      printf("$\t<\t<\t<\taccept\n");
51  }
52
53  void printPredictiveParsingTable() {
54      printf("\nPredictive Parsing Table\n");
55      printf("\tid\t+\t*\t$\n");
56      printf("E\tE->TE'\t\t\t\n");
57      printf("E'\t\tE'->+TE'\t\tE'->ε\n");
58      printf("T\tT->FT'\t\t\t\n");
59      printf("T'\t\tT'->ε\tT'->*FT'\n");
60      printf("F\tF->id\n");
61  }
62
63  int main() {
64      getGrammar();
65
66      if (isOperatorPrecedenceGrammar()) {
67          printLeadingTrailingSets();
68          printOperatorPrecedenceTable();
69      } else {
70          printFirstFollowSets();
71          printPredictiveParsingTable();
72      }
73
74      return 0;
75  }
```



```
main.c
31          printf("Leading(%c): { (,id }\n", productions[i][0]);
32          printf("Trailing(%c): { id,)}\n", productions[i][0]);
33      }
34  }
35
36  void printFirstFollowSets() {
37      printf("\n First and Follow Sets\n");
38      for (int i = 0; i < numProductions; i++) {
39          printf("First(%c): { a }\n", productions[i][0]);
40          printf("Follow(%c): { $ }\n", productions[i][0]);
41      }
42  }
43
44  void printOperatorPrecedenceTable() {
45      printf("\nOperator Precedence Parsing Table\n");
                                                    input
Operator Precedence Parsing Table
        +       *       id      $
+       >       <       <       >
*       >       >       <       >
id      >       >       -       >
$       <       <       <       accept
id

...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT-11

Title: Implementation of Shift-Reduce Parser Lab Activity: WAP to implement Shift-Reduce Parser in C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100
char input[MAX], stack[MAX];
int top = -1, ip = 0;
void printTable(const char *action) {
    printf("%-20s %-20s %-20s\n", stack, input + ip, action);
}
int isReducible() {
    /
    if (top >= 1 && stack[top] == 'i' && stack[top - 1] == 'd')
        {
        stack[top - 1] = 'E';
        top--;
        return 1;
    } else if (top >= 2 && stack[top] == 'E' && stack[top - 1]
        == '+' && stack[top - 2] == 'E') {  // E + E -> E
```

```c
    } else if (top >= 2 && stack[top] == 'E' && stack[top - 1]
        == '+' && stack[top - 2] == 'E') {  // E + E -> E
        top -= 2;
        stack[top] = 'E';
        return 1;
    } else if (top >= 2 && stack[top] == 'E' && stack[top - 1]
        == '*' && stack[top - 2] == 'E') {  // E * E -> E
        top -= 2;
        stack[top] = 'E';
        return 1;
    } else if (top >= 2 && stack[top] == ')' && stack[top - 1]
        == 'E' && stack[top - 2] == '(') {  // (E) -> E
        top -= 2;
        stack[top] = 'E';
        return 1;
    }
    return 0;
}

```

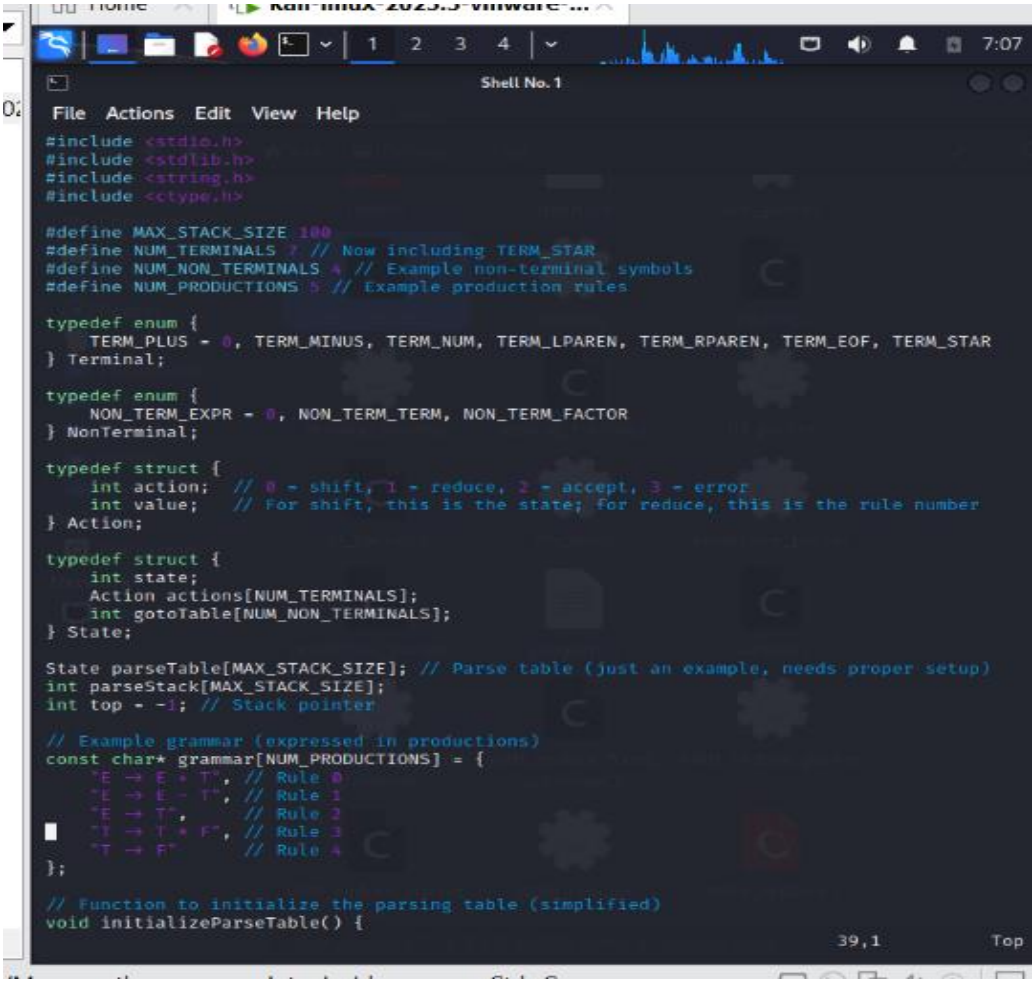
```c
int main() {
    printf("Enter input string (tokenized, end with $):\n> ");
    scanf("%s", input);
    printf("\n%-20s %-20s %-20s\n", "STACK", "INPUT", "ACTION");
    printf("-------------------------------------------------------
        ---------\n");

    while (1) {
        stack[++top] = input[ip++];
        stack[top + 1] = '\0';
            printTable("SHIFT");
        while (isReducible()) {
            printTable("REDUCE");
        }
        if (stack[0] == 'E' && stack[1] == '\0' && input[ip] ==
            '$') {
            printTable("ACCEPTED");
            break;
        }
```

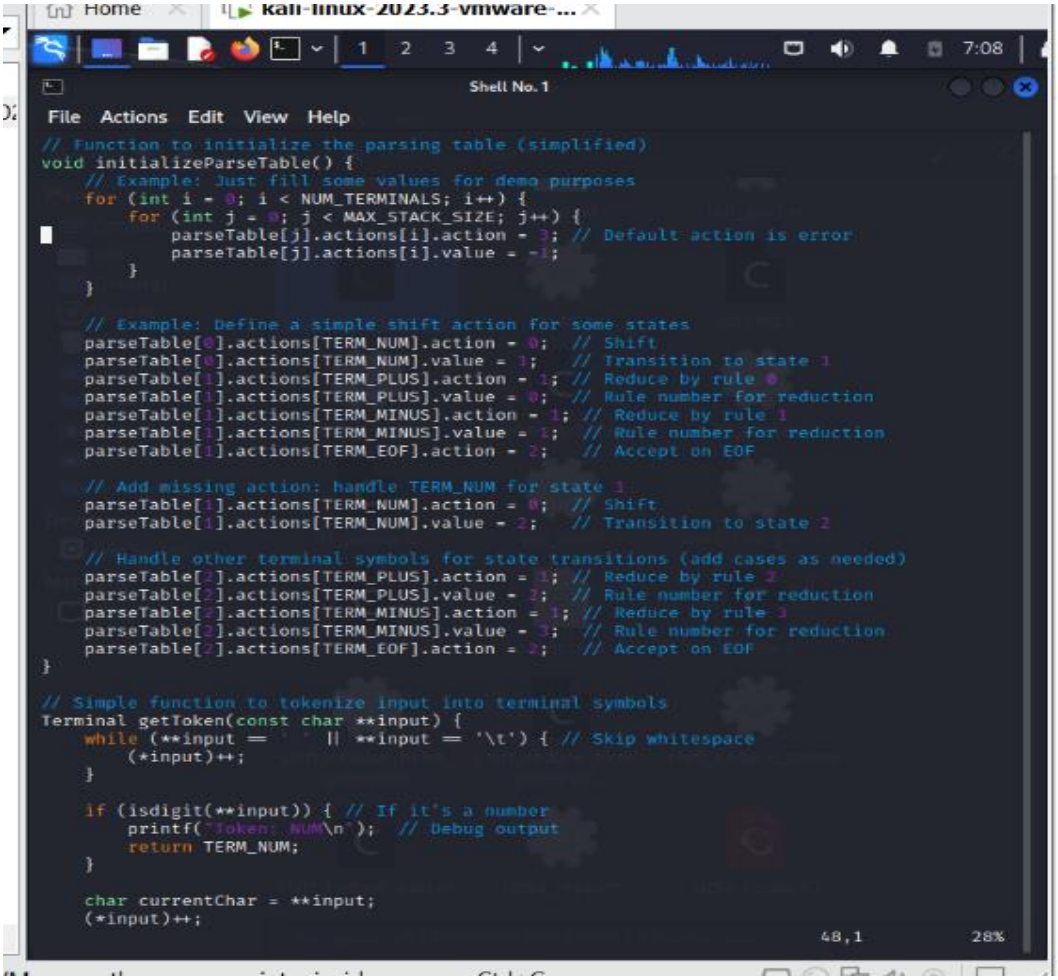
```c
        stack[top + 1] = '\0';
            printTable("SHIFT");
        while (isReducible()) {
            printTable("REDUCE");
        }
        if (stack[0] == 'E' && stack[1] == '\0' && input[ip] ==
            '$') {
            printTable("ACCEPTED");
            break;
        }
        if (input[ip] == '\0') {
            printTable("REJECTED");
            break;
        }
    }

    return 0;
}
```

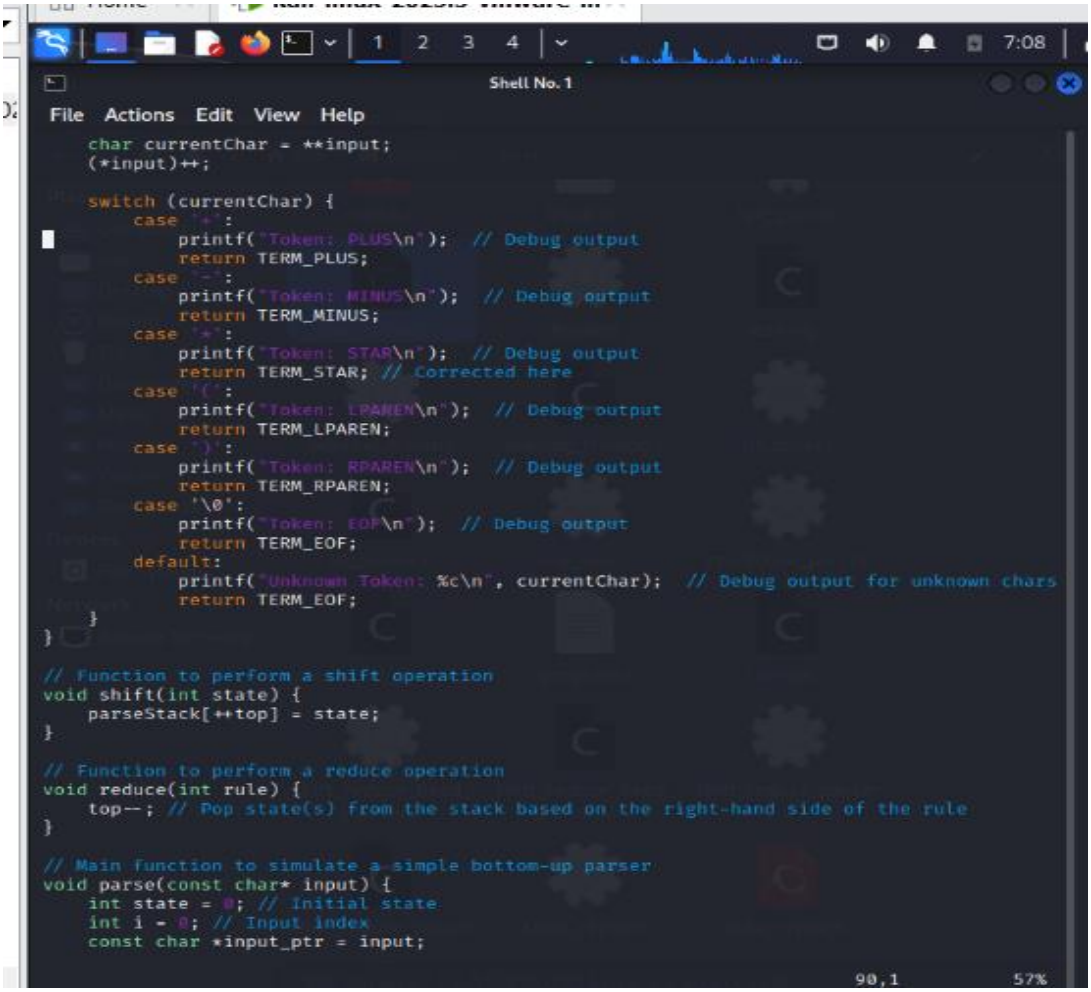
```
Enter input string (tokenized, end with $):
> id+id*id$

STACK                INPUT                ACTION
-------------------------------------------------------------
i                    d+id*id$             SHIFT
id                   +id*id$              SHIFT
id+                  id*id$               SHIFT
id+i                 d*id$                SHIFT
id+id                *id$                 SHIFT
id+id*               id$                  SHIFT
id+id*i              d$                   SHIFT
id+id*id             $                    SHIFT
id+id*id$                                 SHIFT
id+id*id$                                 Accepted
```

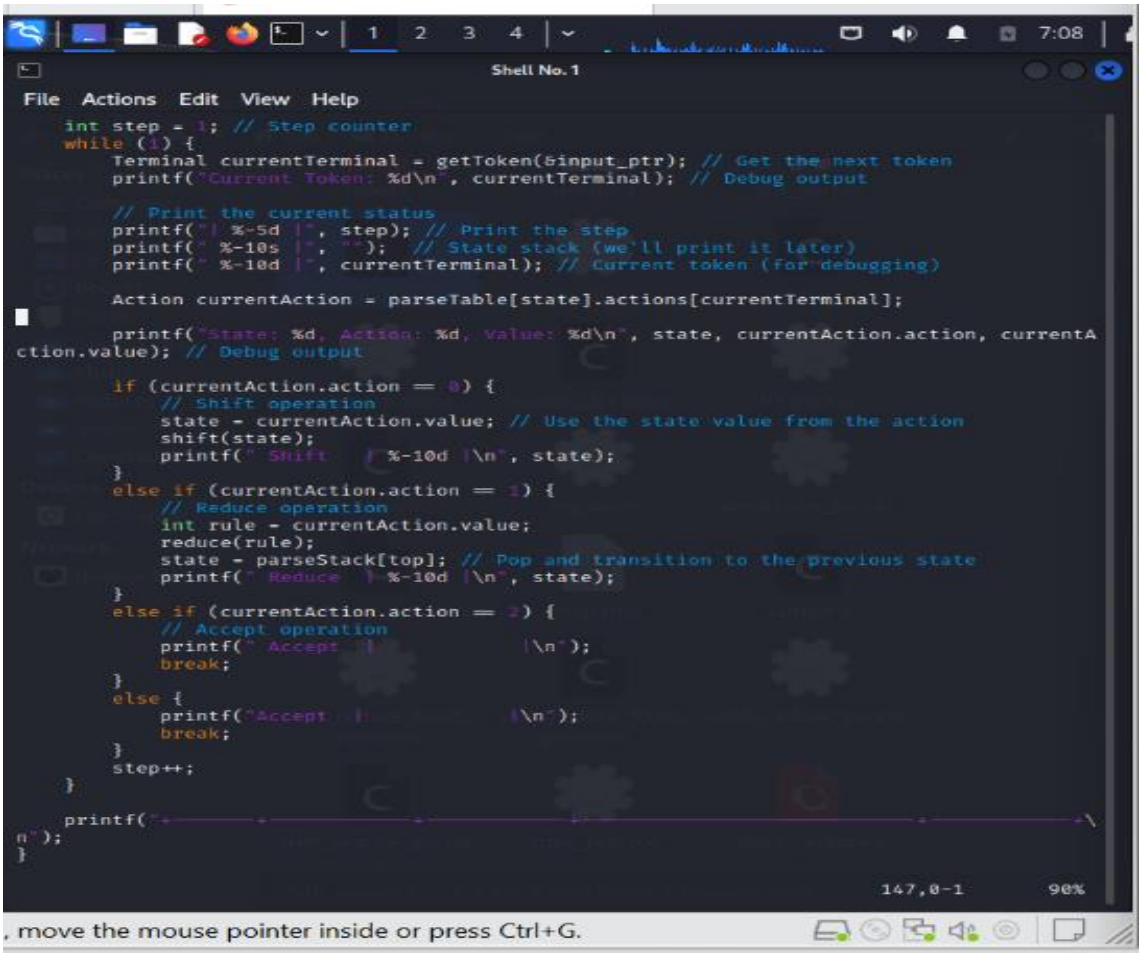
EXPERIMENT-12

Title: Design of a LALR Bottom-Up Parser for a given Language Lab Activity: WAP to derive a LALR Bottom Up Parser for the given language in C

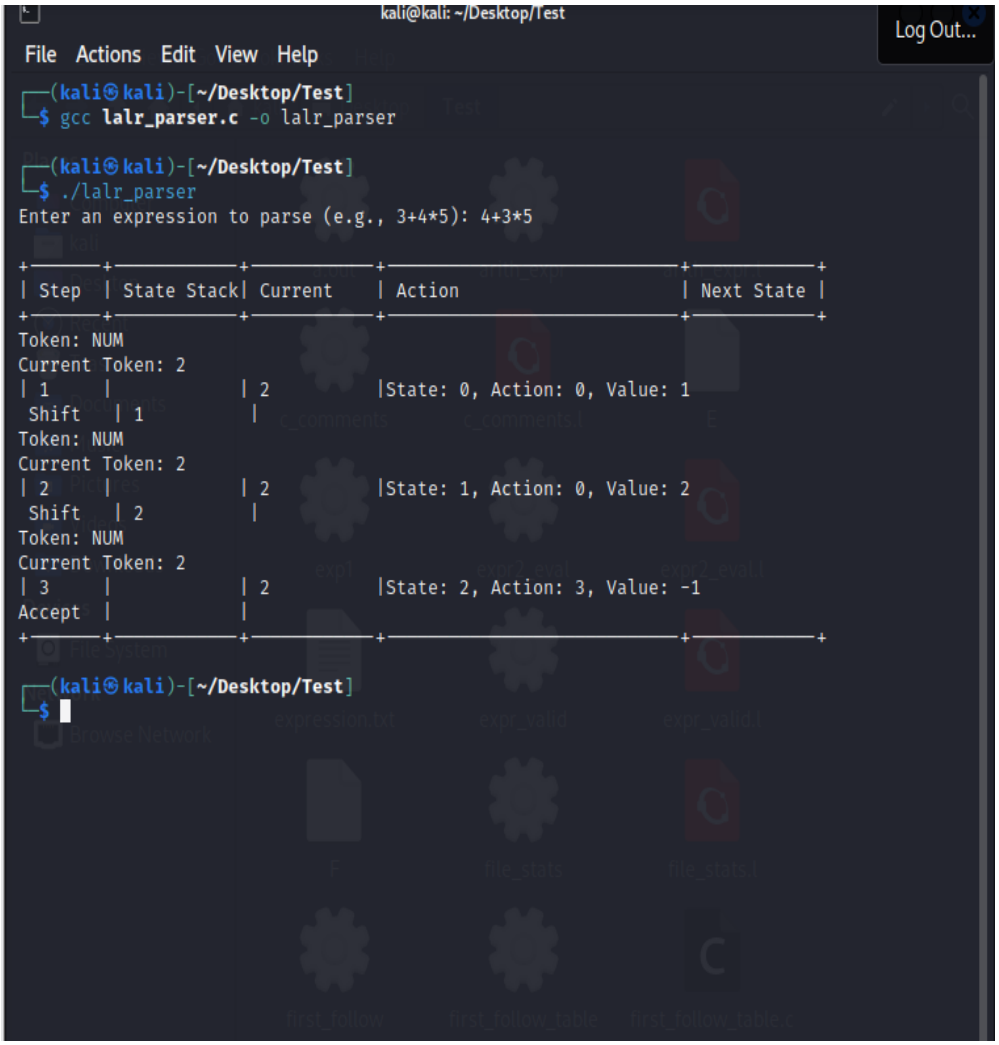



Home
kali-linux-2023.3-vmware-...
1 2 3 4
7:08
Shell No. 1
File Actions Edit View Help
// Function to initialize the parsing table (simplified)
void initializeParseTable() {
// Example: Just fill some values for demo purposes
for (int i = 0; i < NUM_TERMINALS; i++) {
for (int j = 0; j < MAX_STACK_SIZE; j++) {
parseTable[j].actions[i].action = 3; // Default action is error
parseTable[j].actions[i].value = -1;
}
}
// Example: Define a simple shift action for some states
parseTable[0].actions[TERM_NUM].action = 0; // Shift
parseTable[0].actions[TERM_NUM].value = 1; // Transition to state 1
parseTable[1].actions[TERM_PLUS].action = 1; // Reduce by rule 0
parseTable[1].actions[TERM_PLUS].value = 0; // Rule number for reduction
parseTable[1].actions[TERM_MINUS].action = 1; // Reduce by rule 1
parseTable[1].actions[TERM_MINUS].value = 1; // Rule number for reduction
parseTable[1].actions[TERM_EOF].action = 2; // Accept on EOF
// Add missing action: handle TERM_NUM for state 1
parseTable[1].actions[TERM_NUM].action = 0; // Shift
parseTable[1].actions[TERM_NUM].value = 2; // Transition to state 2
// Handle other terminal symbols for state transitions (add cases as needed)
parseTable[2].actions[TERM_PLUS].action = 1; // Reduce by rule 2
parseTable[2].actions[TERM_PLUS].value = 2; // Rule number for reduction
parseTable[2].actions[TERM_MINUS].action = 1; // Reduce by rule 3
parseTable[2].actions[TERM_MINUS].value = 3; // Rule number for reduction
parseTable[2].actions[TERM_EOF].action = 2; // Accept on EOF
}
// Simple function to tokenize input into terminal symbols
Terminal getToken(const char **input) {
while (**input == ' ' || **input == '\t') { // Skip whitespace
(*input)++;
}
if (isdigit(**input)) { // If it's a number
printf("Token: NUM\n"); // Debug output
return TERM_NUM;
}
char currentChar = **input;
(*input)++;
48,1 28%


1 2 3 4
7:08
Shell No. 1
File Actions Edit View Help
char currentChar = **input;
(*input)++;
switch (currentChar) {
case '+':
printf("Token: PLUS\n"); // Debug output
return TERM_PLUS;
case '-':
printf("Token: MINUS\n"); // Debug output
return TERM_MINUS;
case '*':
printf("Token: STAR\n"); // Debug output
return TERM_STAR; // Corrected here
case '(':
printf("Token: LPAREN\n"); // Debug output
return TERM_LPAREN;
case ')':
printf("Token: RPAREN\n"); // Debug output
return TERM_RPAREN;
case '\0':
printf("Token: EOF\n"); // Debug output
return TERM_EOF;
default:
printf("Unknown Token: %c\n", currentChar); // Debug output for unknown chars
return TERM_EOF;
}
}
// Function to perform a shift operation
void shift(int state) {
parseStack[++top] = state;
}
// Function to perform a reduce operation
void reduce(int rule) {
top--; // Pop state(s) from the stack based on the right-hand side of the rule
}
// Main function to simulate a simple bottom-up parser
void parse(const char* input) {
int state = 0; // Initial state
int i = 0; // Input index
const char *input_ptr = input;
90,1 57%


1 2 3 4
7:08
Shell No. 1
File Actions Edit View Help
int step = 1; // Step counter
while (1) {
Terminal currentTerminal = getToken(&input_ptr); // Get the next token
printf("Current Token: %d\n", currentTerminal); // Debug output
// Print the current status
printf("| %-5d |", step); // Print the step
printf(" %-10s |", ""); // State stack (we'll print it later)
printf(" %-10d |", currentTerminal); // Current token (for debugging)
Action currentAction = parseTable[state].actions[currentTerminal];
printf("State: %d, Action: %d, Value: %d\n", state, currentAction.action, currentAction.value); // Debug output
if (currentAction.action == 0) {
// Shift operation
state = currentAction.value; // Use the state value from the action
shift(state);
printf(" Shift | %-10d |\n", state);
}
else if (currentAction.action == 1) {
// Reduce operation
int rule = currentAction.value;
reduce(rule);
state = parseStack[top]; // Pop and transition to the previous state
printf(" Reduce | %-10d |\n", state);
}
else if (currentAction.action == 2) {
// Accept operation
printf(" Accept | |\n");
break;
}
else {
printf("Accept | |\n");
break;
}
step++;
}
printf("+
n");
}
147,0-1 90%
, move the mouse pointer inside or press Ctrl+G.


Shell No. 1
File Actions Edit View Help
state = parseStack[top]; // Pop and transition to the previous state
printf(" Reduce | %-10d |\n", state);
}
else if (currentAction.action == 2) {
// Accept operation
printf(" Accept | |\n");
break;
}
else {
printf("Accept | |\n");
break;
}
step++;
}
printf("+
n");
}
int main() {
initializeParseTable(); // Set up the parse table (to be filled with correct values)
// Get input from the user
char input[100];
printf("Enter an expression to parse (e.g., 3+4*5): ");
fgets(input, sizeof(input), stdin);
input[strcspn(input, "\n")] = '\0'; // Remove newline character if present
parse(input); // Parse the input
return 0;
}


kali@kali: ~/Desktop/Test
Log Out...
File Actions Edit View Help
(kali㉿kali)-[~/Desktop/Test]
$ gcc lalr_parser.c -o lalr_parser
(kali㉿kali)-[~/Desktop/Test]
$ ./lalr_parser
Enter an expression to parse (e.g., 3+4*5): 4+3*5
+------+-------------+---------+-------------------------+------------+
| Step | State Stack | Current | Action | Next State |
+------+-------------+---------+-------------------------+------------+
Token: NUM
Current Token: 2
| 1 | | 2 |State: 0, Action: 0, Value: 1
Shift | 1 |
Token: NUM
Current Token: 2
| 2 | | 2 |State: 1, Action: 0, Value: 2
Shift | 2 |
Token: NUM
Current Token: 2
| 3 | | 2 |State: 2, Action: 3, Value: -1
Accept | |
+------+-------------+---------+-------------------------+------------+
(kali㉿kali)-[~/Desktop/Test]
$