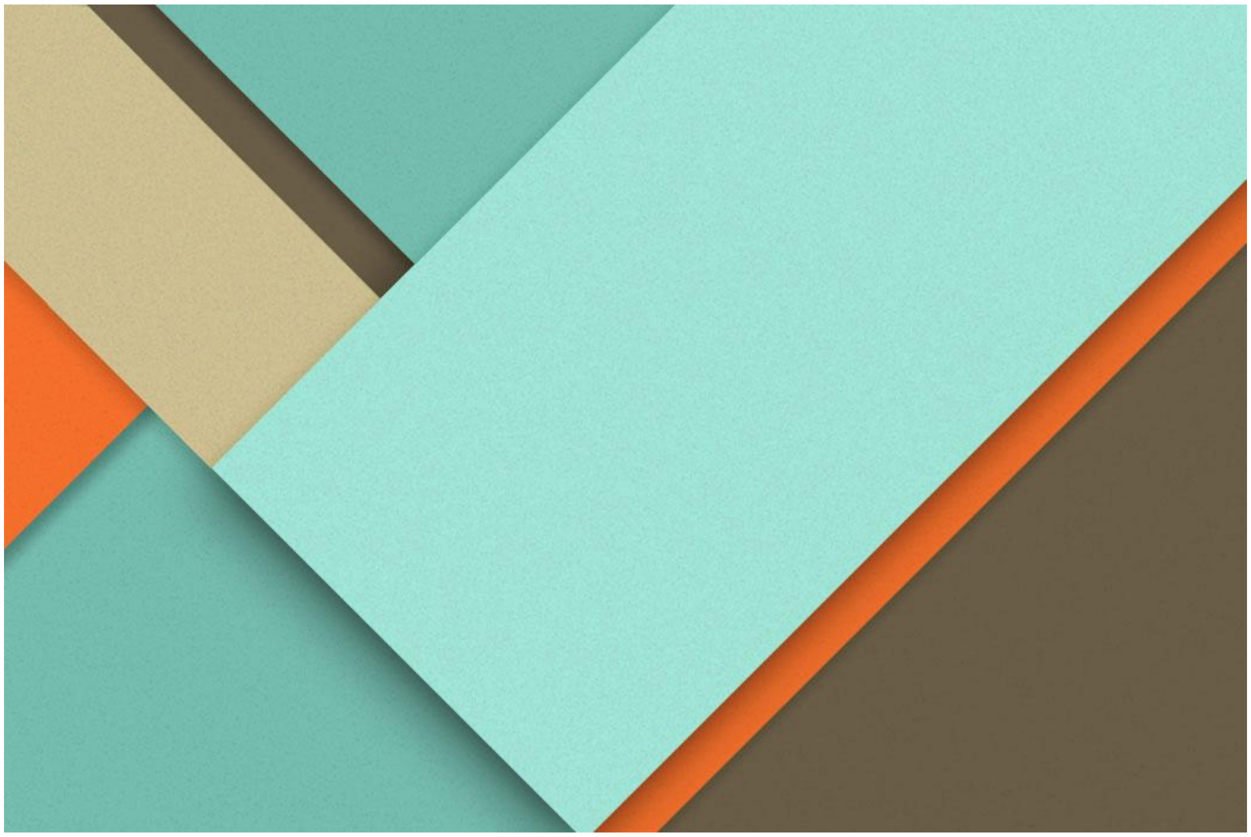


Lex & Yacc Multipurpose Calculator

DSE22.3F - Operating Systems Course Work

—

Group Members

Pramuditha Weerakoon	CODSE22.3F-121
Vihanga Deshan	CODSE22.3F-013
Geemith Devasurendra	CODSE22.3F-134
Ryan Silva	CODSE22.3F-122

Table Of Contents

Introduction

In the realm of software development, we embark on a journey to craft a versatile calculator utilizing the formidable combination of Lex and Yacc. This calculator possesses the prowess to undertake five core mathematical operations: addition, subtraction, multiplication, division, as well as handling more advanced mathematical concepts like square roots, exponentiation, and modulo operations.

But that's not all; this calculator transcends mere arithmetic functionality. It embraces the user's needs with open arms, providing a suite of intuitive commands to assist in problem-solving. The user can invoke commands such as 'h' (help) to seek guidance, and 'q' (exist) to gracefully conclude their calculation session.

By harnessing the power of Lex and Yacc, we aim to forge an application that is both robust and user-friendly, empowering users to perform mathematical feats with ease and grace.

Lex Code

```
%{
#include "calc.tab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

extern double result;

%}

%%

[ \t]+              /* skip whitespace */
[0-9]+(\.[0-9]*)?   { yylval = atof(yytext); return NUMBER; }
"sqrt"              { yylval = 0.0; return SQRT; }
"^"                 { return EXPONENT; } // Recognize '^' as the EXPONENT token
"%"                 { return MODULO; } // Recognize '%' as the MODULO token
\n                  { return EOL; }
.                   { return yytext[0]; }

%%

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

Yacc Code

```
%{
#include <stdio.h>
#include <math.h>
#include <stdlib.h> // Include for exit() function

int yylex(void);
void yyerror(const char* msg);

double result = 0.0;
int syntax_error = 0; // Flag to track syntax errors

void display_instructions(); // Forward declaration

%}

%token NUMBER
%token COMMAND_ERROR // New token for invalid commands
%token EOL // Declare the EOL token
%token SQRT // Declare the SQRT token
%token EXPONENT // Declare the EXPONENT token for '^' operator
%token MODULO // Declare the MODULO token for '%' operator

%%

input: /* empty */
    | input line EOL {
        if (!syntax_error) {
            printf("\033[0;32mResult: %.2lf\n\033[0m", result); // Limit to two decimal points
        }
        printf("===========================================================\n");
        result = 0.0;
        syntax_error = 0; // Reset syntax error flag
        printf("Enter expressions or commands:\n");
    }
    | input command EOL
    | input error EOL { // Handle syntax errors
        syntax_error = 0; // Reset syntax error flag
        yyerrok; // Clear Bison's error flag
    }
    ;

line: expr { result = $1; }
    ;

expr:   NUMBER          { $$ = $1; }
    | expr '+' term    { $$ = $1 + $3; }
    | expr '-' term    { $$ = $1 - $3; }
    | expr '*' term    { $$ = $1 * $3; }
    | expr '/' term    {
        if ($3 == 0) {
            yyerror("Division by zero");
            $$ = 0;
        } else {
            $$ = $1 / $3;
        }
    }
    | SQRT '(' expr ')' { $$ = sqrt($3); } // Handle square root
    | expr EXPONENT term { $$ = pow($1, $3); } // Handle exponentiation
    | expr MODULO term  { $$ = fmod($1, $3); } // Handle modulo operation
    ;
```

```
term:   factor          { $$ = $1; }
    | term '*' factor  { $$ = $1 * $3; }
    | term '/' factor  {
        if ($3 == 0) {
            yyerror("Division by zero");
            $$ = 0;
        } else {
            $$ = $1 / $3;
        }
    }
    ;

factor: NUMBER          { $$ = $1; }
    | '(' expr ')'     { $$ = $2; }
    ;

command: 'h' { display_instructions(); }
    | 'q' { exit(0); }
    | COMMAND_ERROR { yyerror("Invalid command"); }
    ;

%%

void yyerror(const char* msg) {
    static int error_count = 0;

    if (error_count == 0) {
        fprintf(stderr, "\033[1;31mError: %s\033[0m\n", msg);
        syntax_error = 1; // Set syntax_error flag to 1
    }
    error_count++;

    // Optionally, you can add logic here to handle or log errors as needed.
}

void display_instructions() {
    printf("User-Friendly Calculator\n");
    printf("- 'q' to quit\n");
    printf("- 'h' for help (show this message)\n");
    printf("Example expressions:\n");
    printf("\033[32mCorrect:\033[0m  10+20\n");
    printf("\033[31mIncorrect:\033[0m 10 + 20, 10+d, '10+20'\n");
    printf("==============================================================\n");
    printf("Supported operations:\n");
    printf("- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) %% (modulo)\n");
    printf("Example expressions:\n");
    printf("  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 %% 3'\n");
    printf("This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.\n");
    printf("Enter expressions or commands:\n");
}

int main() {
    display_instructions();
    yyparse();
    return 0;
}
```

Brief Code Explanation

Lex Code Explanation :

Lex tool is used to generate lexical analyzers

- **[\t+] :** skips white passes
- **[0-9]+(\.[0-9]*)? :** number which can be an integer or a floating point , the matched number is then converted to a double using **atof** returns as a number token
- **"sqrt" :** when matches the string "sqrt" and returns token SQRT
- **"^" :** when matches the character "^" and returns token EXPONENT
- **"%" :** when matches the character "%" and returns token MODULO
- **\n :** when matches a new line and returns EOL token

Yacc Code Explanation :

Yacc tool is used to generate parses

- **%token :** defines tokens that parser expects from the lex
- **%% - - between - - %% :** includes and defines grammar,rules and error messages
- Example Expression
    - **expr: expr '+' term { $$ = $1 + $3; } :** expression can be a another expression and another term

Compilation & Running

Step 1 .

Run the following commands to generate the lexer and parser files (lex.yy.c, cal.tab.c, and cal.tab.h):

flex calc.l   # Generate lex.yy.c (the lexer source code)

Compiler Report (after running flex calc.l):

Bison parser generates cal.tab.c and cal.tab.h

Compile the Code:

Step 2 .

Now, compile your lexer and parser code along with any additional code. Include the math library (-lm) since you're using math functions:

```
gcc -o calculator calc.tab.c lex.yy.c -lm
```

Compiler Report (after running gcc):

Compilation completed successfully

If there are any compilation errors or warnings, they will be displayed in the compiler report at this stage.

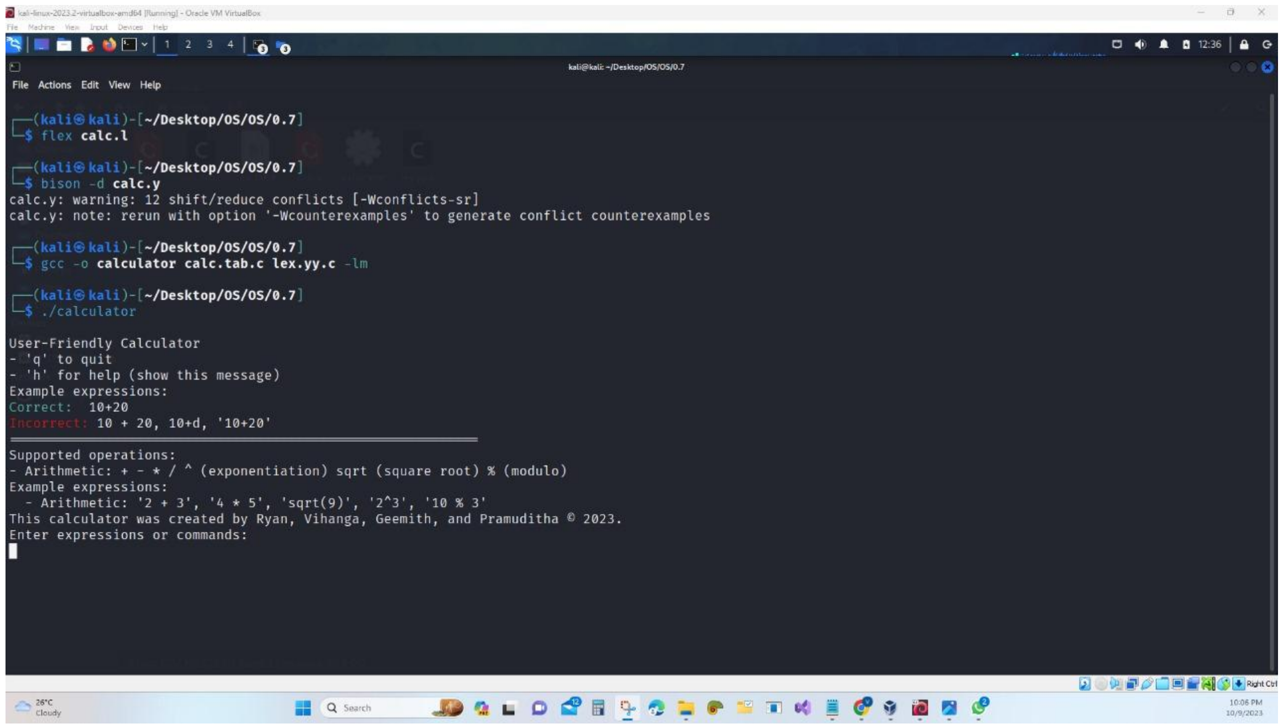
Step 3.

Run the Calculator:

After successful compilation, you can run your calculator using the following command:

./calculator

This will execute your calculator program, and you can start entering expressions or commands.



```
┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ flex calc.l

┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ bison -d calc.y
calc.y: warning: 12 shift/reduce conflicts [-Wconflicts-sr]
calc.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples

┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ gcc -o calculator calc.tab.c lex.yy.c -lm

┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
═══════════════════════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
█
```

Features

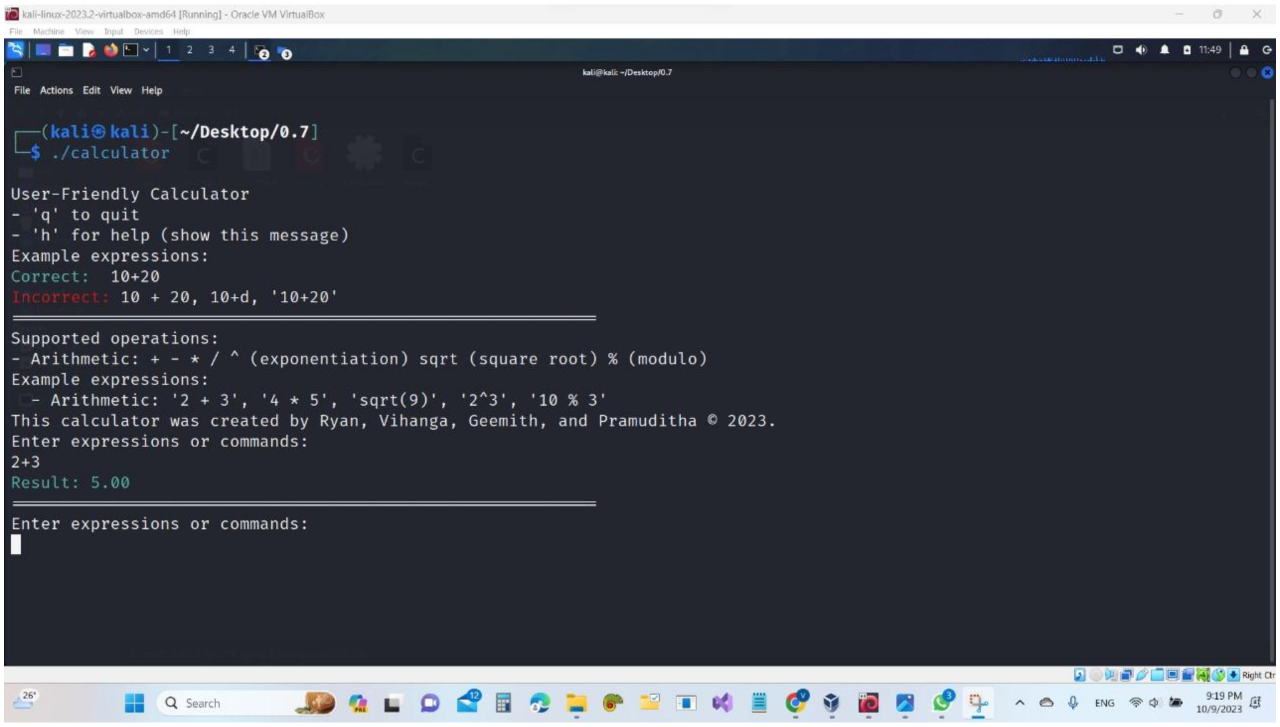
1 Arithmetic Operation

The calculator supports basic arithmetic operations like addition(+) , subtraction(-) ,multiplication(*) and division (/)

1.1 Test Case: Addition (+)

In this test case, we aim to perform a simple addition operation: 2+3. If the code has no syntax errors, it will display the result "5" in green text.

2 + 3 Expected Output: 5

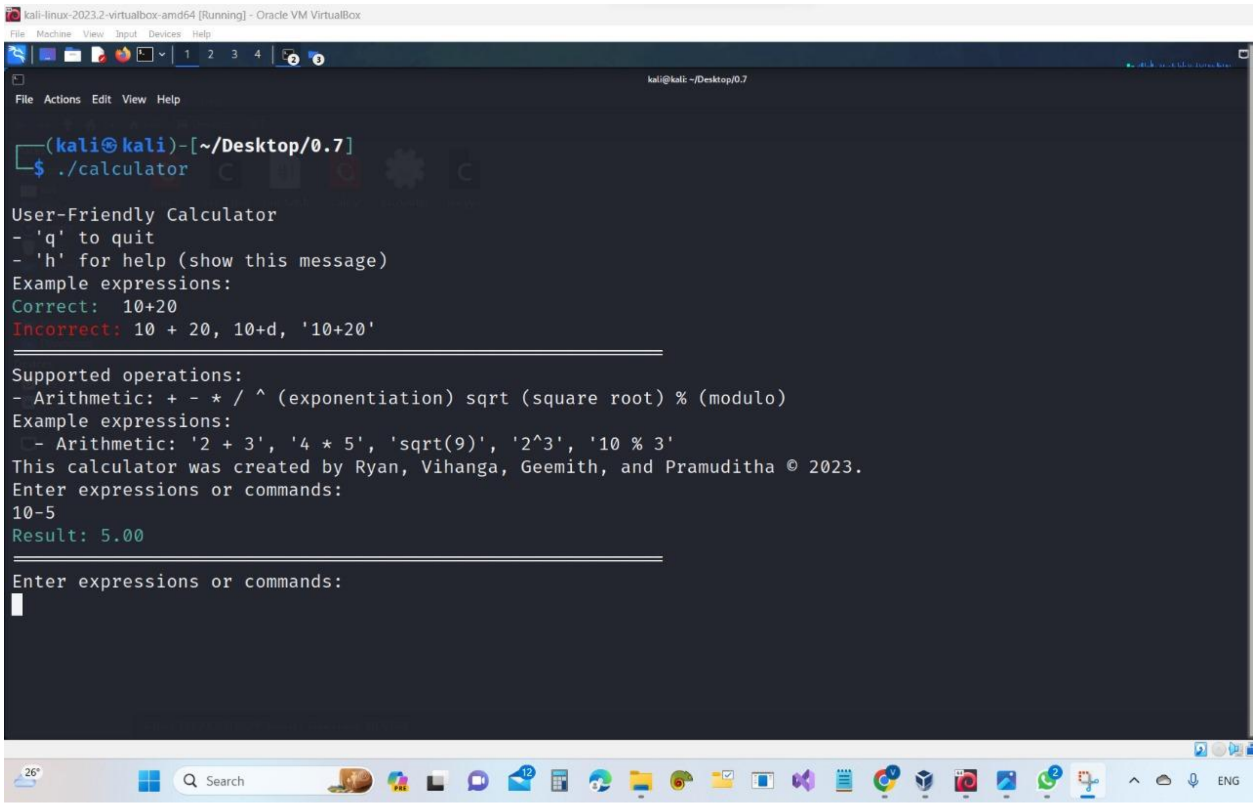


```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
2+3
Result: 5.00
════════════════════════════════════════════════════════════
Enter expressions or commands:
```

1.2 Test Case: Subtraction(-)

In this test case, we aim to perform a simple subtraction operation: 10-5. If the code has no syntax errors, it will display the result "5" in green text.

`10 - 5` Expected Output: 5

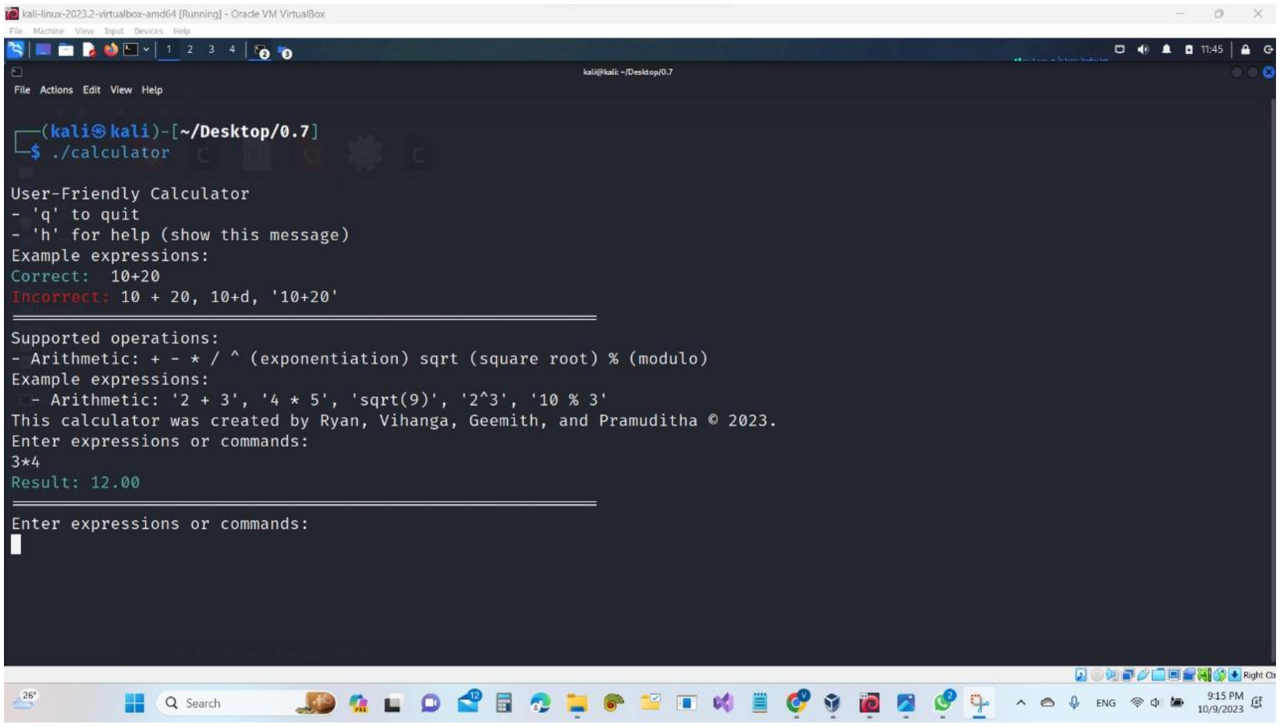

kali-linux-2023.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~/Desktop/0.7
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
══════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
10-5
Result: 5.00
══════════════════════════════════════════════════════════

Enter expressions or commands:
```

1.3 Test Case: Multiplication(*)

In this test case, we aim to perform a simple multiplication operation: 3*4. If the code has no syntax errors, it will display the result "12" in green text.

3 * 4 Expected Output: 12

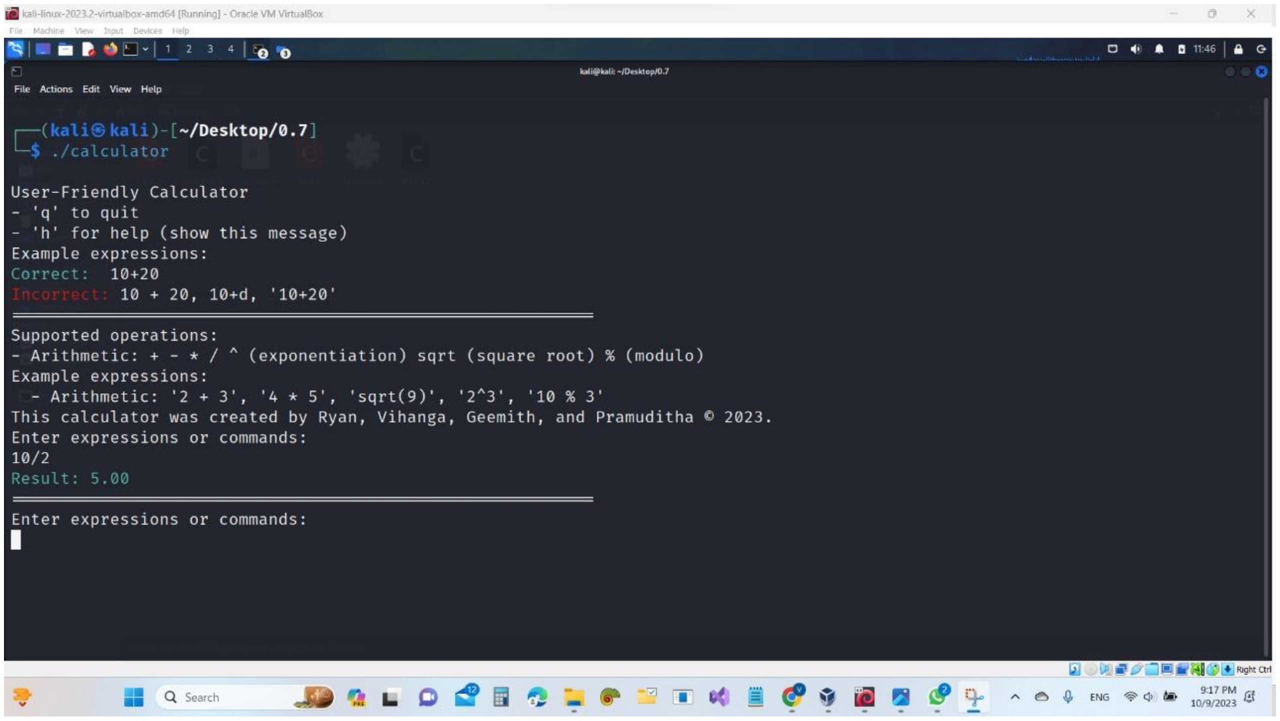


```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
═══════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
3*4
Result: 12.00
═══════════════════════════════════════════════════════════

Enter expressions or commands:
```

1.4 Test Case: Division(/)

In this test case, we aim to perform a simple division operation: 10/2. If the code has no syntax errors, it will display the result "5" in green text.

`10 / 2` Expected Output: 5



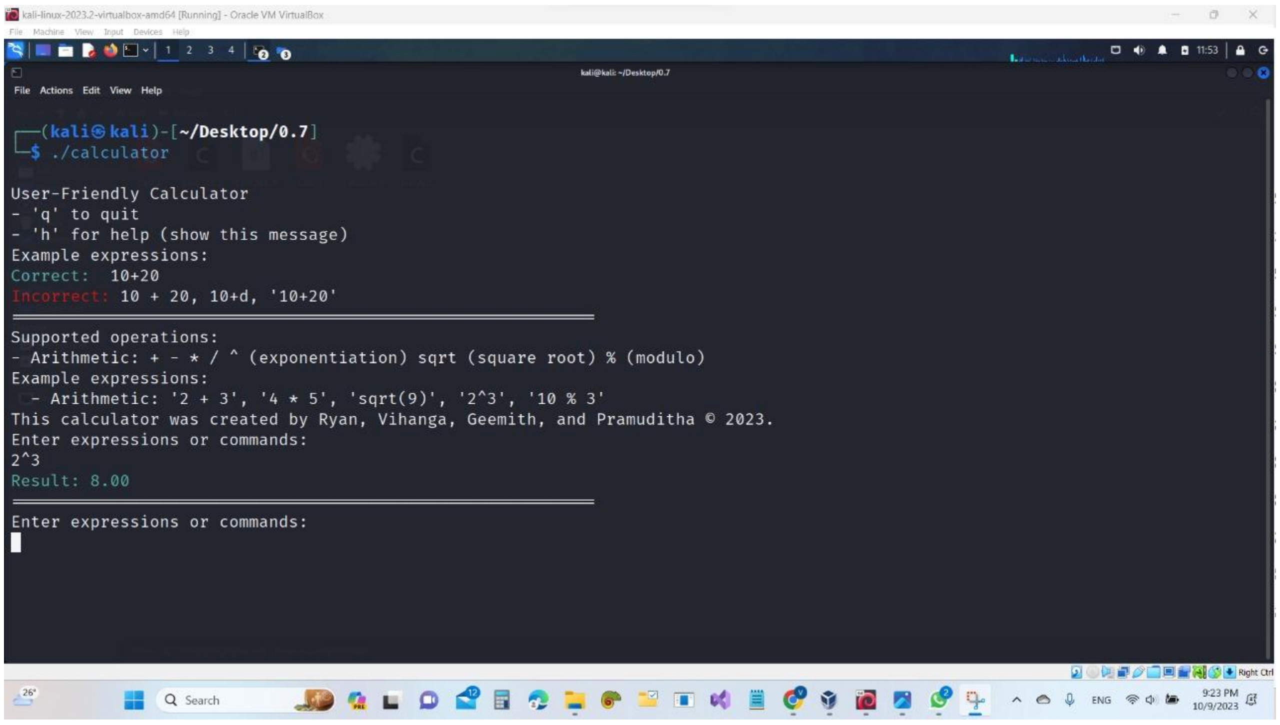
```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
10/2
Result: 5.00
════════════════════════════════════════════════════════════
Enter expressions or commands:
```

2 Exponentiation

The calculator supports Exponentiation to get the power value of the number

2.1 Test Case: Exponentiation(^)

In this test case, we aim to perform 2 ^ 3. If the code has no syntax errors, it will display the result "8" in green text.



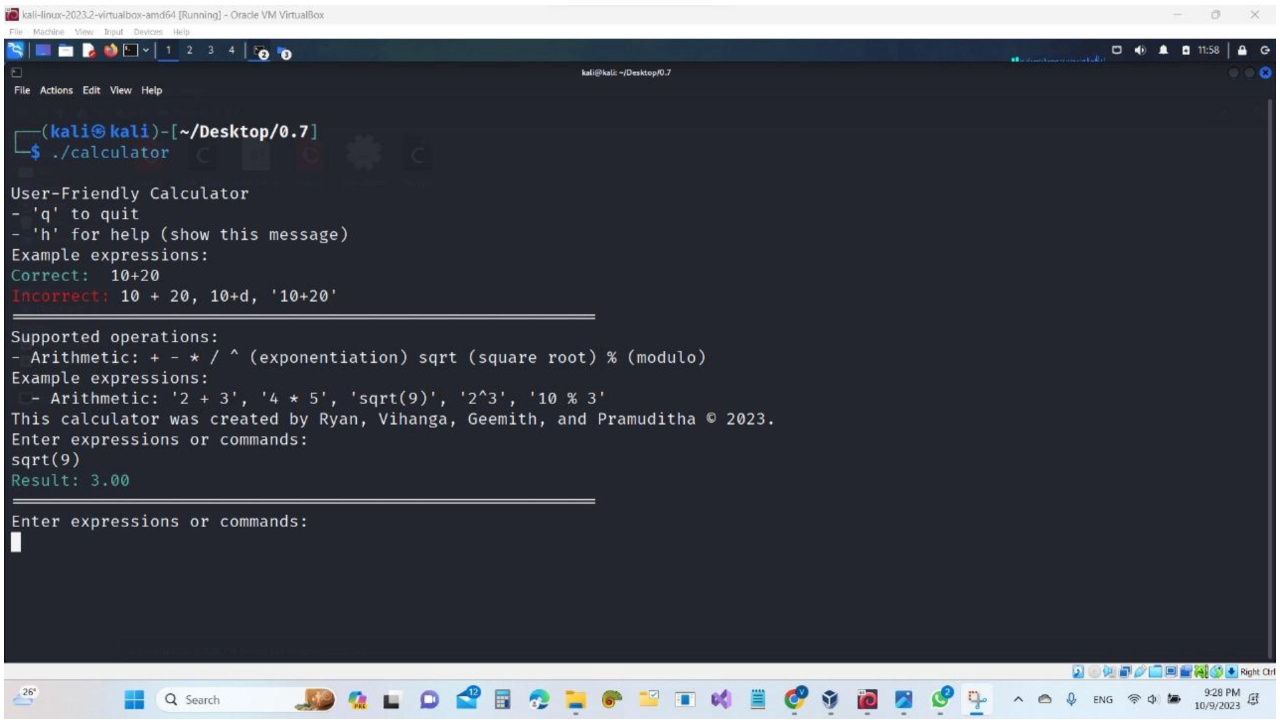
```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
═══════════════════════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
2^3
Result: 8.00
═══════════════════════════════════════════════════════════
Enter expressions or commands:
```

3 SquareRoot

The calculator supports square root of integers a breeze

3.1 Test Case: Square Root(sqrt())

In this test case, we aim to perform sqrt(9). If the code has no syntax errors, it will display the result "3" in green text.



```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
sqrt(9)
Result: 3.00
════════════════════════════════════════════════════════════

Enter expressions or commands:
█
```

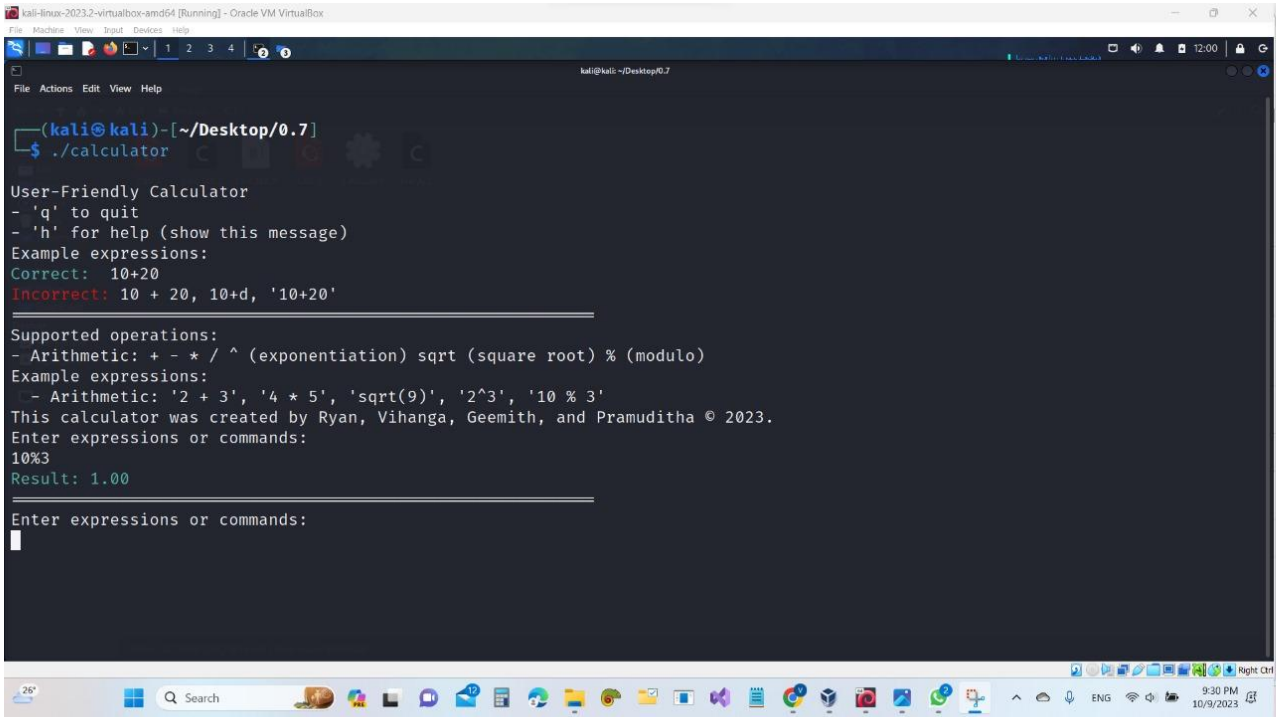
kali-linux-2023.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~/Desktop/0.7
File Actions Edit View Help
11:58
Search
ENG
9:28 PM
10/9/2023
Right Ctrl

4 Modulo Operation

The modulo operation, often denoted as "%," returns the remainder when one number is divided by another.

4.1 Test Case: Modulo Operation(%)

In this test case, we aim to perform 10%3. If the code has no syntax errors, it will display the result "8" in green text.


kali-linux-2023.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~/Desktop/0.7
File Actions Edit View Help

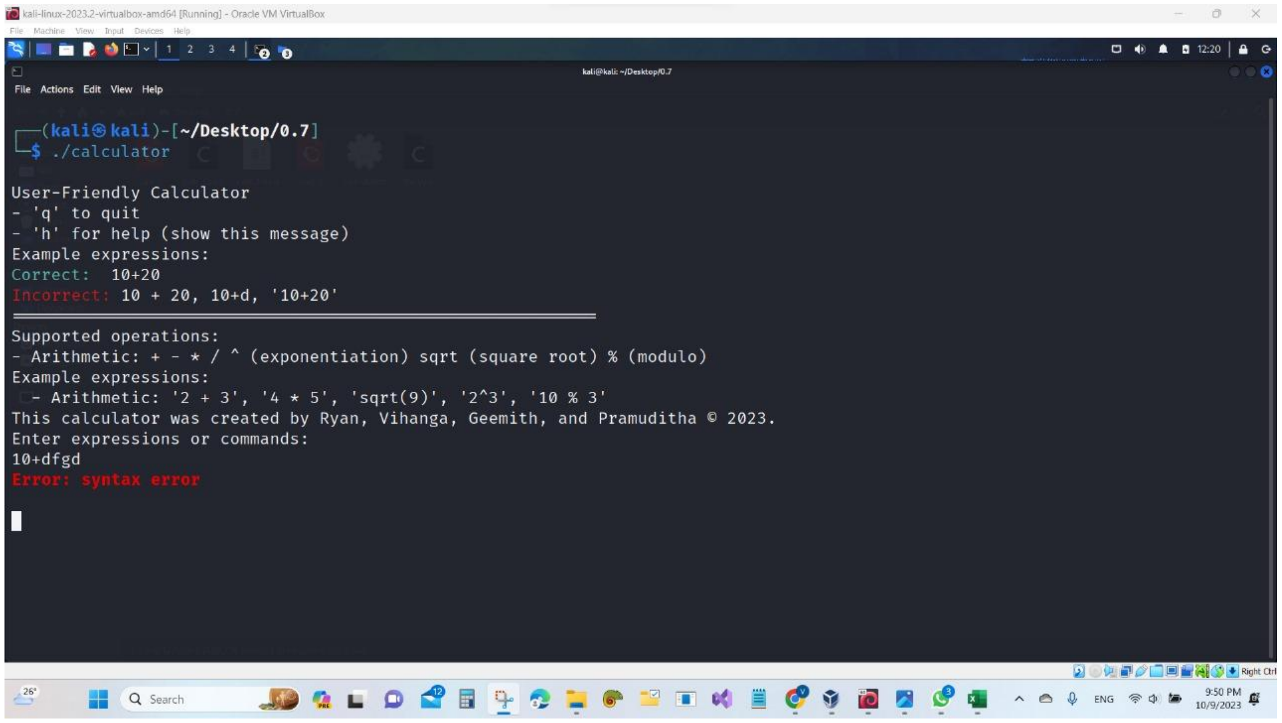
```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
═══════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
10%3
Result: 1.00
═══════════════════════════════════════════════════════════

Enter expressions or commands:
```

5 Error Handling

This calculator is equipped with error handling to ensure precise computations and prevent unexpected issues. It helps provide reliable results even in complex calculations.

5.1 Test Case: Error Handling

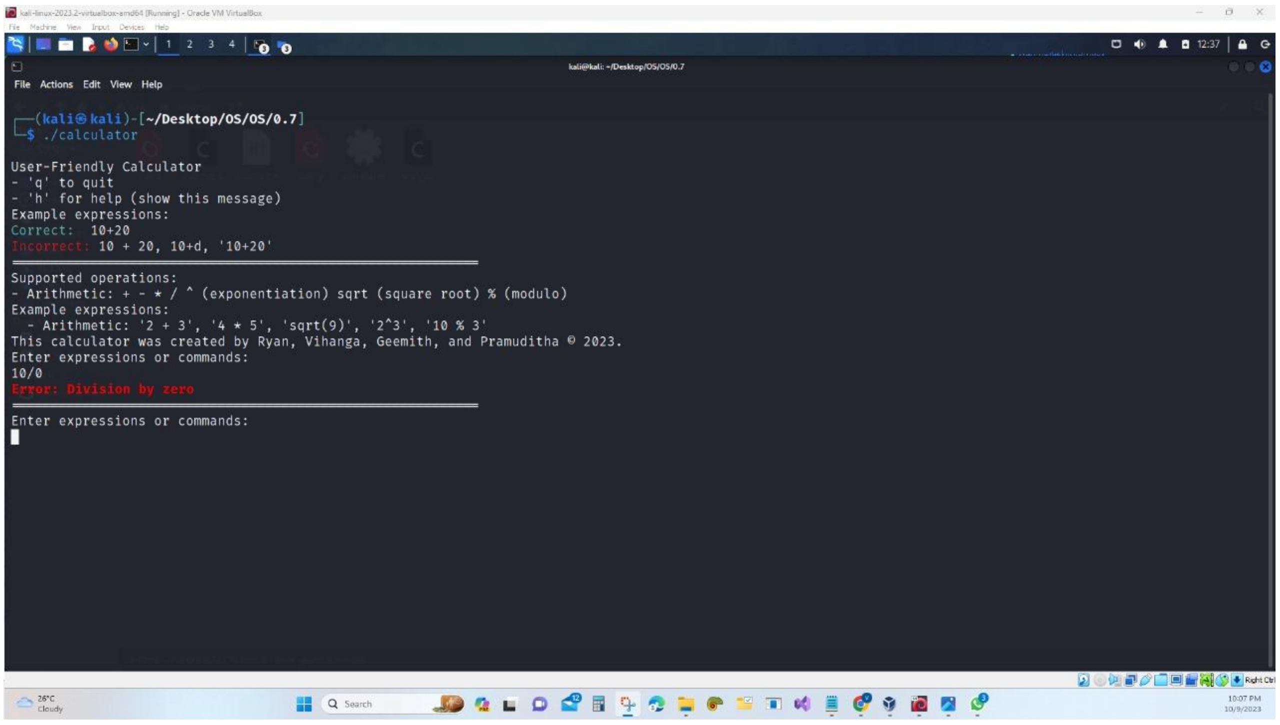
This calculator effectively handles syntax errors to ensure smooth and error-free calculations.In this when you insert letter or any other character that is not Valid in calculator it throws syntax errors



```
┌──(kali㉿kali)-[~/Desktop/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:   10+20
Incorrect: 10 + 20, 10+d, '10+20'
═══════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
10+dfgd
Error: syntax error
```

5.1 Test Case: Divide by zero

This calculator incorporates divide-by-zero protection to prevent mathematical errors and provide meaningful results.



```
┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
10/0
Error: Division by zero
════════════════════════════════════════════════════════════

Enter expressions or commands:
█
```

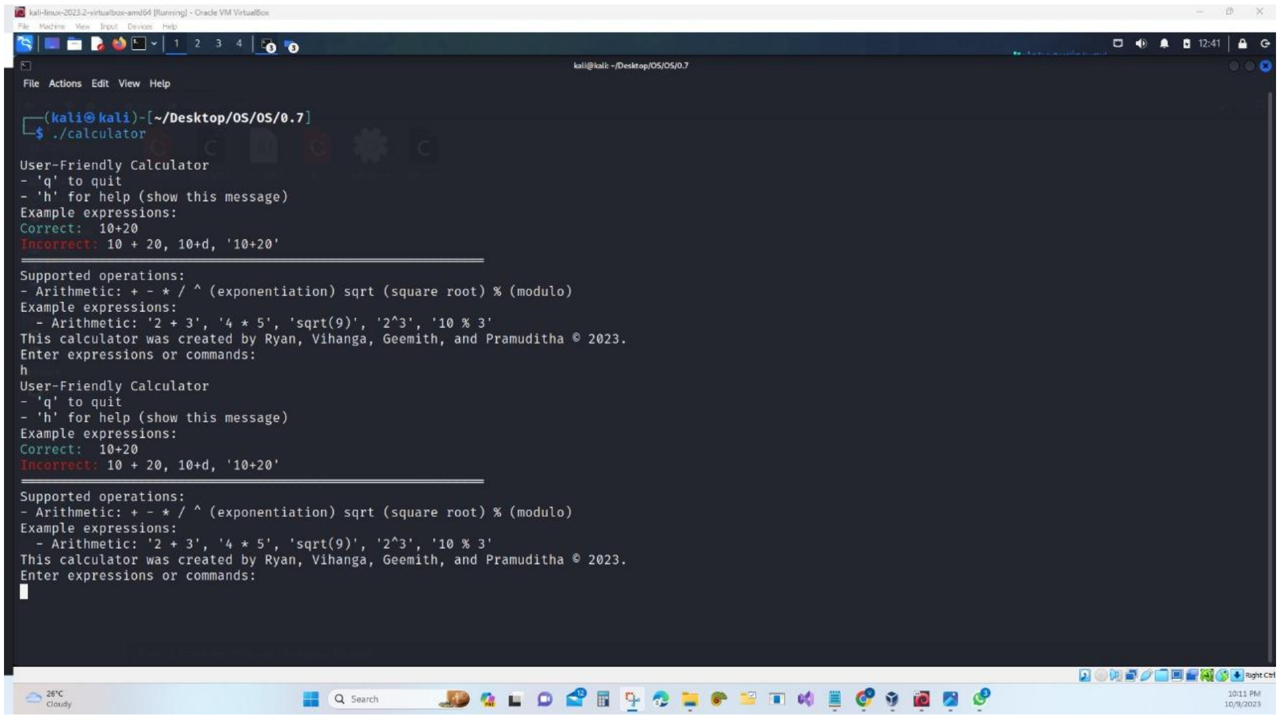
6 User Commands

Pressing "h" activates a help bar, offering assistance to the user.

Pressing "q" closes or quits the program.

6.1 Test Case: Help bar (h)

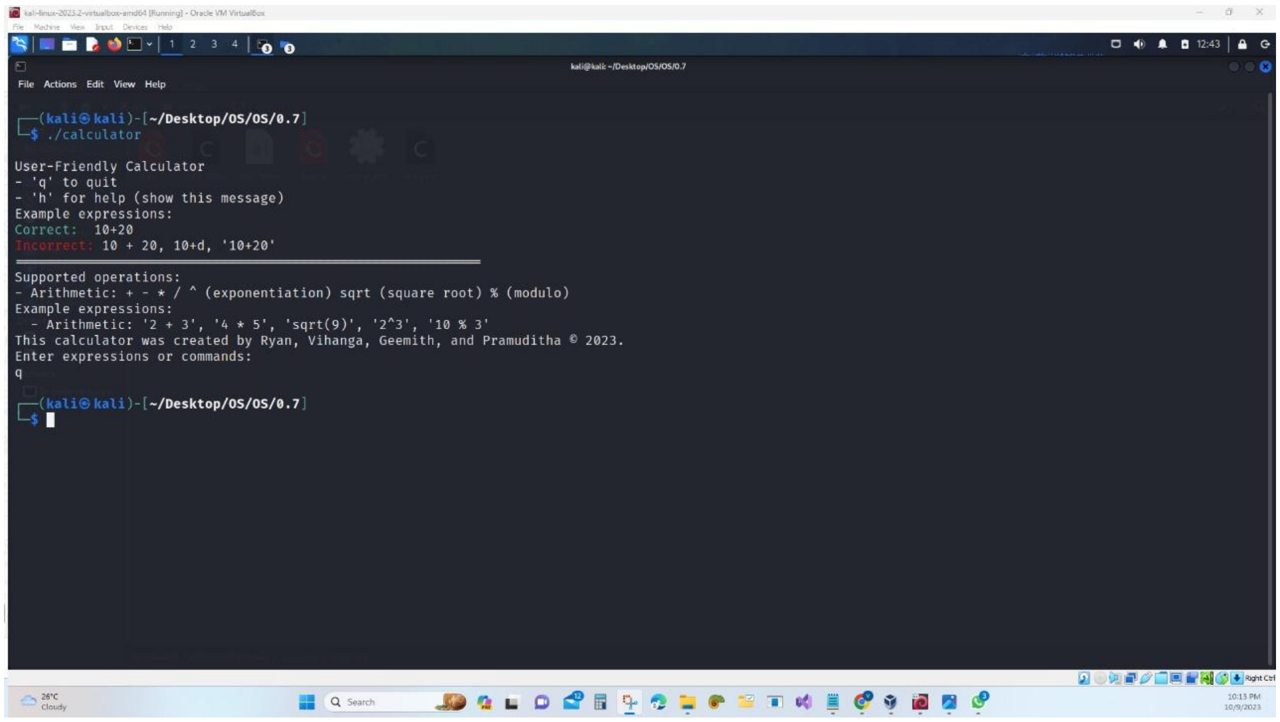
When the "h" key is pressed, the calculator displays a helpful bar, providing guidance and information to the user.



```
┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
h
User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
════════════════════════════════════════════════════════════

Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
█
```

6.2 Test Case: quit (q)

Pressing the "q" key results in the program quitting, allowing the user to exit the calculator.



```
┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ ./calculator

User-Friendly Calculator
- 'q' to quit
- 'h' for help (show this message)
Example expressions:
Correct:  10+20
Incorrect: 10 + 20, 10+d, '10+20'
══════════════════════════════════════════
Supported operations:
- Arithmetic: + - * / ^ (exponentiation) sqrt (square root) % (modulo)
Example expressions:
  - Arithmetic: '2 + 3', '4 * 5', 'sqrt(9)', '2^3', '10 % 3'
This calculator was created by Ryan, Vihanga, Geemith, and Pramuditha © 2023.
Enter expressions or commands:
q

┌──(kali㉿kali)-[~/Desktop/OS/OS/0.7]
└─$ 
```

Conclusion

In conclusion, a multi-purpose calculator program, powered by tools like Lex and Yacc, offers an easy and efficient way to perform both basic and complicated mathematical operations. Lex aids in parsing and breaking down user inputs, while Yacc is responsible for defining the rules and logic to solve mathematical problems. This program is versatile and compatible with a wide range of devices, making math easy and accessible to users from all walks of life. Whether for educational purposes or everyday tasks, such a calculator simplifies calculations, providing users with a valuable tool thanks to the advancements in technology.

Reference

YouTube :- https://youtu.be/M8VVzZODTrc?si=cDYZbRXUDUIF5ZFh

:- https://youtu.be/5ZmFlxrNaN8?si=ck6Lew-ElWfgaZAj

Books :- Lex and Yacc

Websites :- https://stackoverflow.com