Flex and yacc EXAMPLE

```
Enter line: (+ 2b1 4b1)
Syntax OK.
Result: 6.000000
Enter line: (+ 5b1 7b2)
Syntax OK.
Result: 8.500000
Enter line: (+ 3b1 (* 3b1 5b2))
Syntax OK.
Result: 10.500000
Enter line:
```

Lisp Example

```
SATIR: (+ 2b1 3b1)
Result: 5b1
SATIR: (+ 2b1 (* 4b1 3b1))
Result: 14b1
SATIR: (def sum x y (+ x y))
#function
SATIR: (sum 7b1 4b1)
Result: 11b1
SATIR: (+ 2b1  4b1 3b1)
*** - Syntax error Stop Program
```

test.txt

```
1   (+ 2b1 3b1)
2   (+ 2b1 (* 4b1 3b1))
3   (def sum x y (+ x y))
4   (sum 7b1 4b1)
5   (+ 2b1 3b1 4b1)
```

```
Değer girin: (def sum x y z (+ x(+  y z)))
#function
---- Bitiş ----
Değer girin: (sum 2b1 3b1 4b2)
Result: 14b2
---- Bitiş ----
Değer girin:
```

```
---- Bitiş ----
Değer girin: (* 2b1 4b1)
Result: 8b1
---- Bitiş ----
Değer girin: (* 4b1 (- 3b1 2b5))
Result: 52b5
---- Bitiş ----
Değer girin: (exit)

arife@LAPTOP-SRROEDP3:/mnt/c/Users/LENOVO/Desktop/210104004294_Yurtseven_Arife$
```

```
(defun error_function (copy_token index)
```

In this function I identify errors

```
(defun evaluate-math-expression (operator operand1 operand2)
```
I perform separate operations on the numerator and denominator.

```
(defun parse-fraction-string (operand)
```
I collect and collect the operand for the operation to be performed.

```
(defun function-operation-function (token-type token-value)
```
In order to perform the operations, starting from the innermost part, I take the parts that are the operations and equate the number with x and the number with y, that is, equalize the variables and the numerical value.

```
(defun def-function (token-value token-type)
```
I copy my lists and keep the function name somewhere and execute it.