**CIS400 – Project 3**

**The document included:**

* Data Required
* Summary Table
* Testing Screenshot
* Source Code

**Required Data:**  
3 \* +2.5 =   
5 / -3 \* 4 =  
(3 \* 2 ) + 3 =  
5 \*\*3 =   
=  
234.56 + -34.56 \* 4.0 / -2 =  
+1 - -1 =  
--1 + ++1 =  
5.0 / 2 =  
+6.5.5 - +1 =

4+-3.1.3=

+3++=

5/+0.5=

2..=

3\*1+2.5-/2=

.5++1=

0.2+/10.5=

6.10/=

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | Input | Expected result | Pass/ Fail Output |
| Legal Expression | 3 \* +2.5 = | 7.5 | Pass |
| Legal Expression | 5 / -3 \* 4 = | -6.6667 | Pass |
| Illegal Expression | (3 \* 2) + 3 = | “(“not a <digit> | Fail |
| Illegal Expression | 5\*\*3= | “\*” not a <sign> | Fail |
| Illegal Expression | = | Empty String | Fail |
| Legal Expression | 234.56 + -34.56 \* 4.0 / -2= | -400 | Pass |
| Legal Expression | +1 - -1 = | 2 | Pass |
| Illegal Expression | --1+++1 = | “-“ is not a <digit>  “+” is not a <digit> | Fail |
| Legal Expression | 5.0 / 2 = | 2.5 | Pass |
| Illegal Expression | +6.5.5 - +1 = | Role Violated – You can only use 1 point for [.<unsign>] | Fail |
| Illegal Expression | 4+-3.1.3= | Role Violated – You can only use 1 point for [.<unsign>] | Fail |
| Illegal Expression | +3++= | “+” is not an <operation> | Fail |
| Legal Expression | 5/+0.5= | 10 | pass |
| Illegal Expression | 2..= | “.“ is not a <digit> | Fail |
| Illegal Expression | 3\*1+2.5-/2= | “/” is not a <sign> | Fail |
| Illegal Expression | .5++1= | Error, Role Violated | Fail |
| Illegal Expression | 0.2+/10.5= | “/” is not a <sign> | Fail |
| Illegal Expression | 6.10/= | “/” is not an <operation> | Fail |

TEST 1: 3\*+2.5=

Graphical user interface, text, application

Description automatically generated

Test 2: 5/-3\*4=

Graphical user interface, text, application, Word

Description automatically generated

Test3: (3\*2)+3=

Graphical user interface, text, application, Word

Description automatically generated

Test 4: 5\*\*3=

Graphical user interface, text, application

Description automatically generated

Test 5: =

Graphical user interface, application, Word

Description automatically generated

Test 6: ​234.56 + -34.56 \* 4.0 / -2 =

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Test 7: +1—1=

Graphical user interface, text, application, Word

Description automatically generated

Test 8: --1++1=

Graphical user interface, application, Word

Description automatically generated

Test 9: 5.0/2=

Graphical user interface, text, application, Word

Description automatically generated

Test 10: +6.5.*5*-+1=

Graphical user interface, text, application, Word

Description automatically generated

Test 11: 4+-3.1.3=

Graphical user interface, text, application, Word

Description automatically generated

Test 12: +3++ =

Graphical user interface, application, Word

Description automatically generated

Test 13: 5/+0.5=

Graphical user interface, application

Description automatically generated

Test 14: 2..=

Graphical user interface, application, Word

Description automatically generated

Test 15: 3\*1+2.5-/2=

Graphical user interface, application

Description automatically generated

Test 16: .5++1=

Graphical user interface, application, Word

Description automatically generated

Test 17 = 0.2+/10.5

Graphical user interface, application, Word

Description automatically generated

Test 18: 6.10/=

Graphical user interface, application, Word

Description automatically generated

**Source Code:**

#include <iostream>

#include<string>

#include<iomanip>

#include<math.h>

#include<stdlib.h>

#include<fstream>

using namespace std;

ofstream out("C:/Users/zeinabsabra/source/repos/Project3-CIS400/output.txt");

string show = "\t->";

double res = 0;

// calculator

double calculator(float a, float b, char o)

{

double r = 0;

if (o == '+')

r = a + b;

else if (o == '-')

r = a - b;

else if (o == '\*')

r = a \* b;

else

r = a / b;

return r;

}

bool operator1(char op)

{

if (op == '+' || op == '-' || op == '\*' || op == '/')

return true;

else

{

return false;

}

}

bool sign(char s)

{

if (s == '+' || s == '-')

{

return true;

}

else

{

return false;

}

}

bool digit(char d)

{

if (d == '0' || d == '1' || d == '2' || d == '3' || d == '4' || d == '5' || d == '6' || d == '7' || d == '8' || d == '9')

{

return true;

}

else

{

return false;

}

}

bool unsign(string unsign1)

{

bool u = false;

out << show << " <digit>{<digit>}[.<unsigned>][<operator><expression>] =" << endl;

if (unsign1.length() == 0)

{

out << "Error: " << endl;

out << "Rule voilated:\t\t <digit> -> 0|1|2|3|4|5|6|7|8|9" << endl;

return false;

}

for (int i = 0; i < unsign1.length(); i++)

{

u = digit(unsign1.at(i));

if (u == false)

{

out << "Error: \"" << unsign1.at(i) << "\" is not a <digit>." << endl;

out << "Rule voilated:\t\t <digit> -> 0|1|2|3|4|5|6|7|8|9" << endl;

return false;

}

}

return u;

}

bool value(string value1)

{

string dig;

int point;

if (sign(value1.at(0)))

{

out << show << " <sign><unsigned>[.<unsigned>][<operator><expression>] =" << endl;

show = show + " " + value1.at(0);

out << show << " <unsigned>[.<unsigned>][<operator><expression>] =" << endl;

point = value1.find('.');

if (point != string::npos && point != (value1.length() - 1))

{

dig = value1.substr(1, point - 1);

if (unsign(dig))

{

show = show + " " + dig;

out << show << " [.<unsigned>][<operator><expression>] =" << endl;

if (unsign(value1.substr(point + 1)))

{

show += " ." + value1.substr(point + 1);

out << show << " [<operator><expression>] =" << endl;

return true;

}

else

{

return false;

}

}

else

return false;

}

else

{

out << show << " <unsigned>[<unsigned>][<operator><expression>] =" << endl;

if (unsign(value1.substr(1)))

{

show += " " + value1.substr(1);

out << show << " [<operator><expression>] =" << endl;

return true;

}

else

return false;

}

}

else

{

out << show << " [<sign>]<unsigned>[.<unsigned>][<operator><expression>] =" << endl;

out << show << " <unsigned>[.<unsigned>][<operator><expression>] =" << endl;

point = value1.find('.');

if (point != std::string::npos && point != (value1.length() - 1))

{

dig = value1.substr(0, point);

if ((unsign(dig)))

{

show += " " + dig;

out << show << " [.<unsigned>][<operator><expression>] =" << endl;

if (unsign(value1.substr(point + 1)))

{

show += " ." + value1.substr(point + 1);

out << show << " [<operator><expression>] =" << endl;

return true;

}

else

return false;

}

else

return false;

}

else

{

if ((unsign(value1)))

{

show += " " + value1;

out << show << " [<operator><expression>] =" << endl;

return true;

}

else

return false;

}

}

}

bool expression(string exp)

{

char o = '?';

float op1 = 0, op2 = 0;

bool x;

string val;

for (int j = 0; j < exp.length(); j++)

{

if (!digit(exp.at(j)) && exp.at(j) != '.')

{

if (j == 0)

{

if (sign(exp.at(j)))

{

val += exp.at(j);

}

else

{

out << "Error: \"" << exp.at(j) << "\" is not a <sign>." << endl;

out << "Rule voilated:\t\t <sign> -> +|-" << endl;

return false;

}

}

else if (j == exp.length() - 1)

{

out << "Error: \"" << exp.at(j) << "\" is not an <operator>." << endl;

out << "Rule voilated:\t\t <operator> -> +|-|\*|/" << endl;

return false;

}

else if (operator1(exp.at(j - 1)))

{

if (sign(exp.at(j)))

{

val += exp.at(j);

}

else

{

out << "Error: \"" << exp.at(j) << "\" is not a <sign>." << endl;

out << "Rule voilated:\t\t <sign> -> +|-" << endl;

return false;

}

}

else

{

if (operator1(exp.at(j)))

{

out << show << " <value>[<operator><expression>] =" << endl;

x = value(val);

if (x == true)

{

if (o != '?')

{

op2 = stof(val);

res = calculator(op1, op2, o);

o = exp.at(j);

op1 = res;

}

if (operator1(exp.at(0)) && operator1(exp.at(1)))

{

out << "Error: \"" << exp.at(j) << "\" is not a <digit>." << endl;

out << "Rule voilated:\t\t <digit> -> 0|1|2|3|4|5|6|7|8|9" << endl;

return false;

}

else

{

op1 = stof(val);

o = exp.at(j);

}

out << show << " <operator><expression> =" << endl;

show = show + " " + exp.at(j);

out << show << " <expression> =" << endl;

}

}

else

{

out << "Error: \"" << exp.at(j) << "\" is not an <operator>." << endl;

out << "Rule voilated:\t\t <operator> -> +|-|\*|/" << endl;

}

val = "";

}

}

else

{

val += exp.at(j);

}

}

out << show << " <value>[<operator><expression>] =" << endl;

x = value(val);

if (x == true)

{

op2 = stof(val);

res = calculator(op1, op2, o);

cout << "result " << res;

show += " =";

out << show << endl;

}

return x;

}

int main()

{

char c;

string language = "";

bool a = true;

/\* while (a)

{

cout << "enter language:";

cin >> language;

if (language.at(language.length() - 1) == '=')

{

language.pop\_back();

a = expression(language);

if (a == true)

cout << "valid number!" << endl;

else

cout << "invalid number!" << endl;

}

}\*/

if (out.is\_open())

{

cout << "please enter the input string:(end it with '=' sign)" << endl;

while (cin >> c)

{

language += c;

if (c == '=')

goto calculation;

}

calculation:

if (language.at(0) == '=')

{

cout << "Error! Empty String"<< endl;

}

/\*else if (!(language.find('=')))

{

out << "Error! expresion should end with =" << endl;

}\*/

else

{

language.pop\_back();

out << endl << "User input:" << "\t" << language << "=" << endl;

out << endl << "derivaion:" << endl << "<calculation> -> <expression> = " << endl;

a = expression(language);

if (a == true)

out << "Result: " << language << "=" << res << endl;

else

out << "Result: " << language << " \"invalid calculation!\"" << endl;

}

}

else

cout << "file not opened!" << endl;

out.close();

}