CS205 C/ C++ Program Design

Assignment 2

name:刘仁杰 SID：11911808

Part 1. Source Code

```
#include <iostream>
#include <string>
#include <cmath>
#include <vector>
#include <sstream>
#include<algorithm>

using namespace std;

bool isNum(string s);

double multiply(double double1, double double2);

vector<string> init(string str, vector<string> vars, vector<double> value);

bool priorThan(string a, string b);

double stringToNum(string str);

string scan(string str);

vector<double> calculate(vector<string> str1);

vector<string> transform(vector<string> elements);

bool judge(char element);

int searchVar(string str, vector<string> vars);

bool searchEqual(string str);

int main() {
    cout << "Please write your calculation below: " << endl;
    cout << "(input # to exit)" << endl;
    string str;
    getline(cin, str);
    vector<string> vars;
    vector<double> value;
    while (str != "#") {
        vector<string> elements;
        vector<string> str1;
        str.erase(remove(str.begin(),str.end(),' '),str.end());
        //筛选赋值语句
        while (searchEqual(str)) {
            for (int i = 0; i < str.size(); ++i) {
                if (str.at(i) == '=') {
                    int position = searchVar(str.substr(0, i), vars);
                    if (position != -1) {
                        elements = init(str.substr(i + 1, str.size()), vars,
value);
                        str1 = transform(elements);
                        value.at(position) = calculate(str1).at(0);
                        getline(cin, str);
                        break;
                    } else {
                        vars.push_back(str.substr(0, i));
                        elements = init(str.substr(i + 1, str.size()), vars,
value);
                        str1 = transform(elements);
                        value.push_back(calculate(str1).at(0));
                        getline(cin, str);
                        break;
                    }
                }
            }
        }
        elements = init(str, vars, value);
        //判断用户输入并进行数据初始化操作，分解数据并赋值给elements
        while (elements.empty()) {
            getline(cin, str);
            elements = init(str, vars, value);
        }
        str1 = transform(elements);
        cout << "Reverse Polish Notation: ";
        for (const string &string1 : str1) {
            cout << string1 << " ";
        }
        cout << endl;
        cout << "result: " << calculate(str1).at(0) << endl;
        getline(cin, str);
    }
}

bool searchEqual(string str) {
    for (int i = 0; i < str.size(); ++i) {
        if (str.at(i) == '=') {
            return true;
        }
    }
    return false;
}

//将中缀表达式转换为逆波兰表达式
vector<string> transform(vector<string> elements) {
    vector<string> str1;
    vector<string> str2;
    for (int i = 0; i < elements.size(); ++i) {
        if (isNum(elements.at(i))) {
            //如果为数字，则按顺序存储在str1当中
            str1.push_back(elements.at(i));
        } else {
            if (str2.empty() || priorThan(elements.at(i), str2[str2.size() - 1])
|| str2[str2.size() - 1] == "(") {
```

```
                //如果不为数字，先判断其与str2头元素的优先级大小，如果优先级高，则顺序排列，
若优先级低，则反序输出str2中的元素，然后将其存入
                str2.push_back(elements.at(i));
            } else {
                long times = 0;
                for (long j = str2.size() - 1; j >= 0; --j) {
                    if (!priorThan(elements.at(i), str2.at(j)) && str2.at(j) !=
"(" && str2.at(j) != ")") {
                        str1.push_back(str2.at(j));
                        str2.pop_back();
                    } else {
                        if ((str2.at(j) == ")" || times != 0) && str2.at(j) != "
(") {
                            if (str2.at(j) == ")") {
                                str2.pop_back();
                                times++;
                            } else {
                                str1.push_back(str2.at(j));
                            }
                        } else if (str2.at(j) == "(" && times != 0) {
                            times--;
                            str2.pop_back();
                        } else {
                            break;
                        }
                    }
                }
                str2.push_back(elements.at(i));
            }
        }
    }
    //完成逆波兰表达式的最后一步，将str2剩下的元素倒序赋值给str1
    if (!str2.empty()) {
        for (long i = str2.size() - 1; i >= 0; i--) {
            if (str2.at(i) != "(" && str2.at(i) != ")") {
                str1.push_back(str2.at(i));
            }
        }
    }
    return str1;
}

//开始计算，遍历str1
vector<double> calculate(vector<string> str1) {
    vector<double> nums;
    for (int i = 0; i < str1.size(); ++i) {
        double temp = 0;
        if (isNum(str1.at(i))) {
            nums.push_back(stringToNum(str1.at(i)));
        } else {
            //检索到运算符，在这里对nums的头两个元素做运算
            if (str1.at(i) == "+") {
                temp = nums.at(nums.size() - 1) + nums.at(nums.size() - 2);
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            } else if (str1.at(i) == "-") {
                temp = nums.at(nums.size() - 2) - nums.at(nums.size() - 1);
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            } else if (str1.at(i) == "*") {
                temp = multiply(nums.at(nums.size() - 1), nums.at(nums.size() - 
2));
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            } else if (str1.at(i) == "/") {
                temp = nums.at(nums.size() - 2) / nums.at(nums.size() - 1);
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            } else if (str1.at(i) == "^") {
                temp = pow(nums.at(nums.size() - 2), nums.at(nums.size() - 1));
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            } else if (str1.at(i) == "%") {
                temp = fmod(nums.at(nums.size() - 2), nums.at(nums.size() - 1));
                nums.pop_back();
                nums.pop_back();
                nums.push_back(temp);
            }
        }
    }
    return nums;
}

double stringToNum(string str) {
    istringstream iss(str);
    double num;
    iss >> num;
    return num;
}

//将表达式中的特殊函数转换为数值,预处理阶段
string scan(string str) {
    string string1;
    double temp;
    for (int j = 0; j < str.size(); ++j) {
        if (str.at(j) == 'p' && str.at(j + 1) == 'i' && judge(str.at(j + 2))) {
            str.replace(j, 2, to_string(M_PI));
            j++;
        } else if (str.at(j) == 'e' && judge(str.at(j + 1))) {
            str.replace(j, 1, to_string(M_E));
            j++;
        }
    }
    for (int i = 0; i < str.size(); ++i) {
        if (str.at(i) == 's' && str.at(i + 1) == 'i' && str.at(i + 2) == 'n') {
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = sin(stringToNum(str.substr(i + 4, j - i - 4)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'c' && str.at(i + 1) == 'o' && str.at(i + 2) ==
's') {
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = cos(stringToNum(str.substr(i + 4, j - i - 4)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 't' && str.at(i + 1) == 'a' && str.at(i + 2) ==
'n') {
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = tan(stringToNum(str.substr(i + 4, j - i - 4)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 's' && str.at(i + 1) == 'q' && str.at(i + 2) ==
'r' && str.at(i + 3) == 't') {
            for (int j = i + 4; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = sqrt(stringToNum(str.substr(i + 5, j - i - 5)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'e' && str.at(i + 1) == 'x' && str.at(i + 2) ==
'p') {
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = exp(stringToNum(str.substr(i + 4, j - i - 4)));
                    cout << temp << endl;
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'a' && str.at(i + 1) == 'b' && str.at(i + 2) ==
'a') {
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = abs(stringToNum(str.substr(i + 4, j - i - 4)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'l' && str.at(i + 1) == 'g') {
            for (int j = i + 2; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = log10(stringToNum(str.substr(i + 3, j - i - 3)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'l' && str.at(i + 1) == 'n') {
            for (int j = i + 2; j < str.size(); ++j) {
                if (str.at(j) == ')') {
                    temp = log(stringToNum(str.substr(i + 3, j - i - 3)));
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else if (str.at(i) == 'l' && str.at(i + 1) == 'o' && str.at(i + 2) ==
'g') {
            double num1 = 0, num2 = 0;
            int a = 0;
            for (int j = i + 3; j < str.size(); ++j) {
                if (str.at(j) == '(') {
                    num1 = log(stringToNum(str.substr(i + 3, j - i - 3)));
                    a = j;
                }
                if (str.at(j) == ')') {
                    num2 = log(stringToNum(str.substr(a + 1, j - a - 1)));
                    temp = num2 / num1;
                    if (temp < 0) {
                        string1.append('(' + to_string(temp) + ')');
                    } else {
                        string1.append(to_string(temp));
                    }
                    i = j;
                    break;
                }
            }
        } else {
            string1.append(1, str.at(i));
        }
    }
    return string1;
}

bool judge(char element) {
    if (element == '+' || element == '-' || element == '*' || element == '/' ||
element == '^' || element == '%' ||
        element == '(' || element == ')' || element == '.' || (element >= 48 &&
element <= 57) || element == ' ') {
        return true;
    }
    return false;
}

//将字符串转化为vector容器
vector<string> init(string str, vector<string> vars, vector<double> value) {
    str = scan(str);
    char element;
    vector<string> elements;
    for (int i = 0; i < str.size(); i++) {
        element = str[i];
        if (judge(element)) {
            if ((element == '-' || element == '+') && str[i - 1] == '(') {
                elements.emplace_back("0");
                elements.push_back(str.substr(i, 1));
            } else if (element == ' ') {
                continue;
            } else if ((element >= 48 && element <= 57) || element == '.') {
                if (i == str.size() - 1) {
                    elements.push_back(str.substr(i, 1));
                    break;
                } else {
                    for (int j = i + 1; j < str.size(); ++j) {
                        if (j == str.size() - 1 && str.at(j) >= 48 && str.at(j)
<= 57) {
                            elements.push_back(str.substr(i, str.size() - i));
                            i = j;
                        } else if ((str[j] < 48 || str[j] > 57) && str[j] !=
'.') {
                            elements.push_back(str.substr(i, j - i));
                            i = j - 1;
                            break;
                        } else if (j == str.size() - 1 && (str.at(j) < 48 &&
str.at(j) > 57)) {
                            cout << "wrong input!" << endl << "Please input
again: " << endl;
                            elements.clear();
```

```cpp
                                return vector<string>(0);
                            }
                        }
                    }
                } else {
                    elements.push_back(str.substr(i, 1));
                }
            } else {
                if (element == '#') {
                    exit(0);
                } else {
                    int position = searchVar(str.substr(i, str.size()), vars);
                    if (position != -1) {
                        for (int j = i; j < str.size(); ++j) {
                            if (vars.at(position) == str.substr(i, j + 1 - i)) {
                                i = j;
                            }
                        }
                        elements.push_back(to_string(value.at(position)));
                    } else {
                        cout << "wrong input!" << endl << "Please input again: " <<
endl;
                        elements.clear();
                        return vector<string>(0);
                    }
                }
            }
        }
    return elements;
}

int searchVar(string str, vector<string> vars) {
    int position = -1;
    for (int i = 0; i < str.size(); ++i) {
        if (judge(str.at(i))) {
            for (int j = 0; j < vars.size(); ++j) {
                if (vars.at(j) == str.substr(0, i)) {
                    position = j;
                    return position;
                }
            }
        } else if (i == str.size() - 1) {
            for (int j = 0; j < vars.size(); ++j) {
                if (vars.at(j) == str) {
                    position = j;
                    return position;
                }
            }
        }
    }
    return position;
}

bool isNum(string s) {
    stringstream sin(s);
    double t;
    char p;
    if (!(sin >> t))
```

```
        //sin>>t表示把sin转换成double的变量（其实对于int和float型的都会接收），如果转换成
功，则值为非0，如果转换不成功就返回为0
        return false;
    if (sin >> p)
        //检测错误输入中，数字加字符串的输入形式（例如：34.f）
        return false;
    else
        return true;
}

bool priorThan(string a, string b) {
    if (a == "(" || a == ")") {
        return true;
    } else if (a == "^" && b != "(" && b != ")" && b != "^") {
        return true;
    } else if ((a == "*" || a == "/" || a == "%") && (b == "+" || b == "-")) {
        return true;
    } else {
        return false;
    }
}

//任意精度乘法
double multiply(double double1, double double2) {
    string a = to_string(double1);
    string b = to_string(double2);
    int *arr1 = new int[a.size()]();
    int *arr2 = new int[b.size()]();
    int *result = new int[a.size() + b.size()]{};
    int dot1 = 0;
    int dot2 = 0;
    int num = 0;
    string str;
    for (int i = 0; i < a.size(); ++i) {
        if (a.at(i) == '.') {
            dot1 = a.size() - 1 - i;
        }
    }
    for (int i = 0; i < b.size(); ++i) {
        if (b.at(i) == '.') {
            dot2 = b.size() - 1 - i;
        }
    }
    if (a.at(0) == '-') {
        a.erase(remove(a.begin(), a.end(), '-'), a.end());
        num++;
    }
    if (b.at(0) == '-') {
        b.erase(remove(b.begin(), b.end(), '-'), b.end());
        num++;
    }
    a.erase(remove(a.begin(), a.end(), '.'), a.end());
    b.erase(remove(b.begin(), b.end(), '.'), b.end());
    for (int i = 0; i < a.size(); i++) {
        if (a[i] < '0' || a[i] > '9') {
            cout << "your input is wrong!" << endl;
            return 0;
        }
```

```cpp
        arr1[i] = a[i] - '0';
    }
    for (int i = 0; i < b.size(); i++) {
        if (b[i] < '0' || b[i] > '9') {
            cout << "your input is wrong!" << endl;
            return 0;
        }
        arr2[i] = b[i] - '0';
    }
    for (int i = a.size() - 1; i >= 0; i--) {
        for (int j = b.size() - 1; j >= 0; j--) {
            result[a.size() - 1 - i + b.size() - 1 - j] += (arr1[i] * arr2[j]) % 10;
            if (result[a.size() - 1 - i + b.size() - 1 - j] >= 10) {
                result[a.size() - 1 - i + b.size() - j] += result[a.size() - 1 - i + b.size() - 1 - j] / 10;
                result[a.size() - 1 - i + b.size() - 1 - j] = result[a.size() - 1 - i + b.size() - 1 - j] % 10;
            }
            result[a.size() - 1 - i + b.size() - j] += (arr1[i] * arr2[j]) / 10;
            if (result[a.size() - 1 - i + b.size() - j] >= 10) {
                result[a.size() - i + b.size() - j] += result[a.size() - 1 - i + b.size() - j] / 10;
                result[a.size() - 1 - i + b.size() - j] = result[a.size() - 1 - i + b.size() - j] % 10;
            }
        }
    }
    if (result[a.size() + b.size() - 1] != 0) {
        for (int i = a.size() + b.size() - 1; i >= 0; i--) {
            str.append(to_string(result[i]));

        }
    } else {
        for (int i = a.size() + b.size() - 2; i >= 0; i--) {
            str.append(to_string(result[i]));
        }
    }
    for (long i = str.size() - 1; i >= 0; --i) {
        if (str.size() - 1 - i == dot1 + dot2) {
            str.insert(i + 1, 1, '.');
        }
    }
    if (num == 1) {
        str.insert(0, 1, '-');
    }
    return stringToNum(str);
}
```

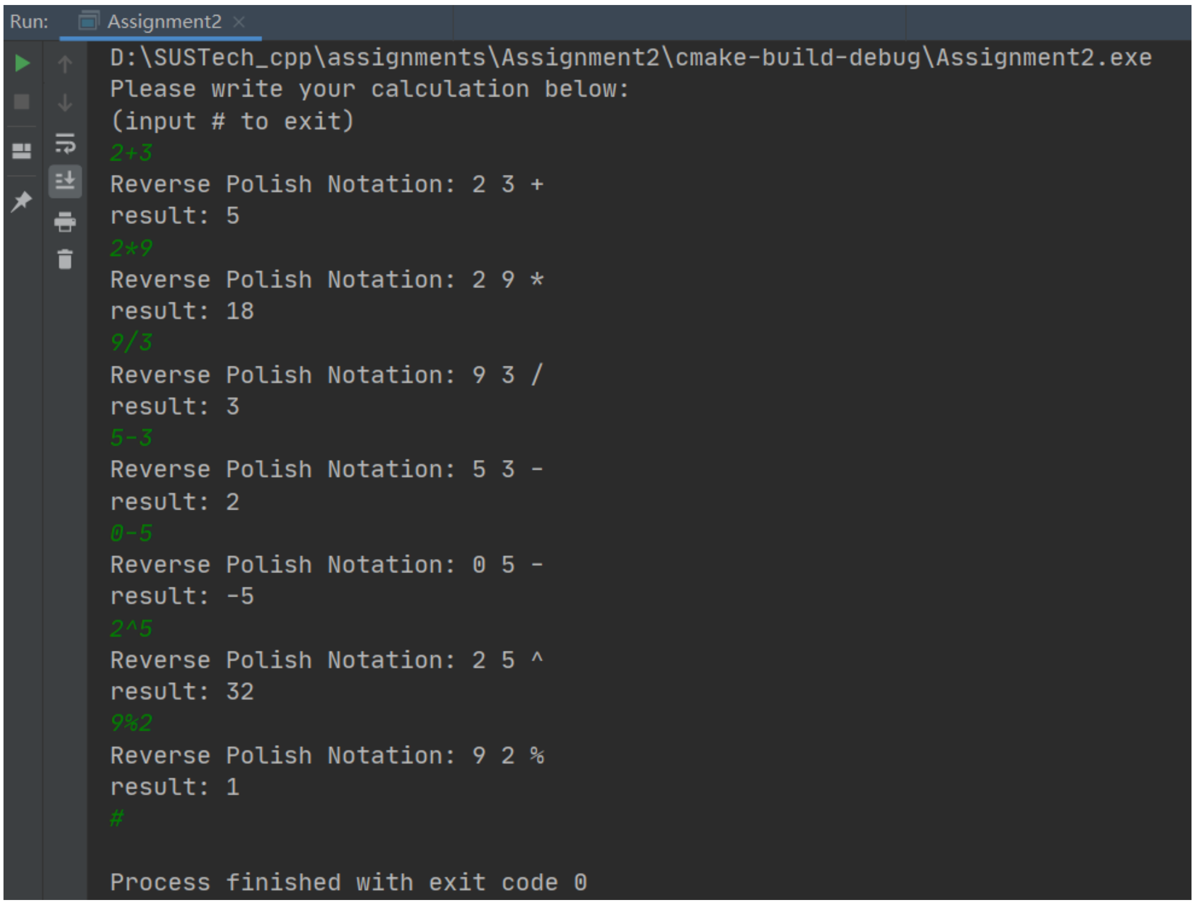
Part 2. Result & Verification

Test case #1(easy calculation(完成简单的运算，包括加减乘除乘方求余)):

input: some basic calculations including add, minus, multiply, divide, power, modulus

output: corresponding results

Screen-short for case #1:



Run: Assignment2

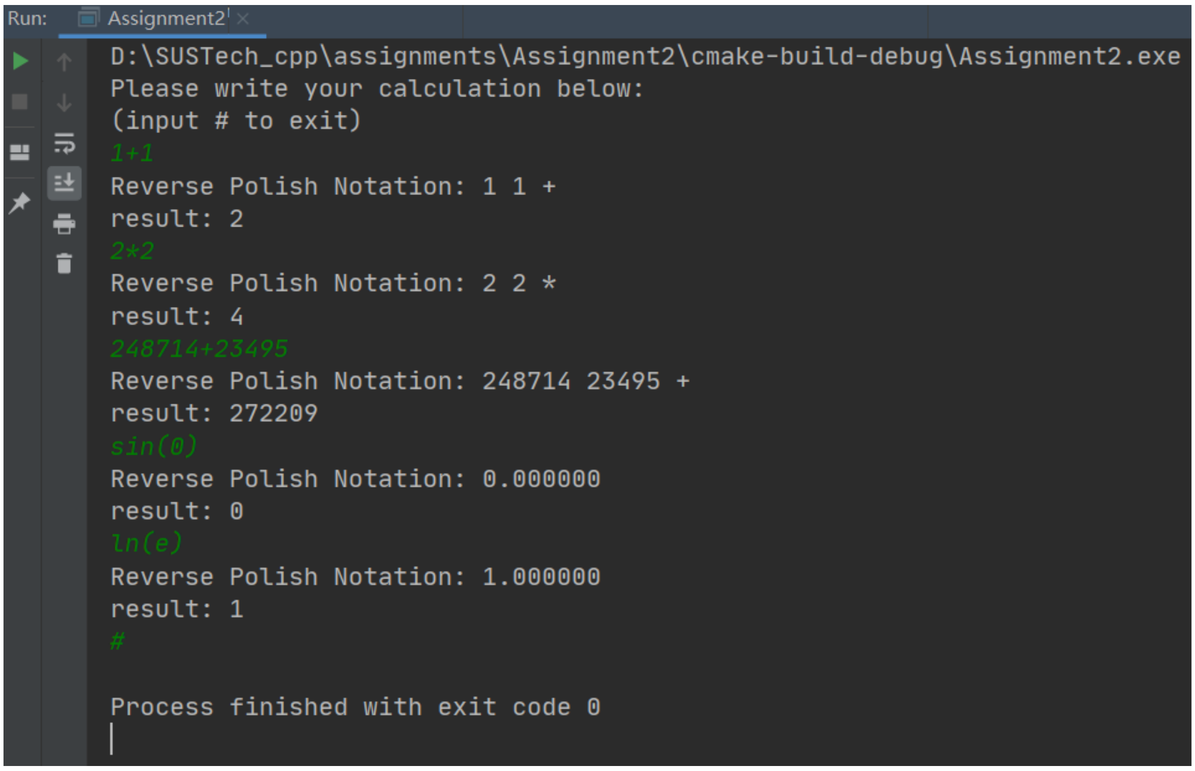
```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
2+3
Reverse Polish Notation: 2 3 +
result: 5
2*9
Reverse Polish Notation: 2 9 *
result: 18
9/3
Reverse Polish Notation: 9 3 /
result: 3
5-3
Reverse Polish Notation: 5 3 -
result: 2
0-5
Reverse Polish Notation: 0 5 -
result: -5
2^5
Reverse Polish Notation: 2 5 ^
result: 32
9%2
Reverse Polish Notation: 9 2 %
result: 1
#

Process finished with exit code 0
```

Test case #2(accept multiple input(一次运行，多次输入, #号退出)):

input: multiple input

output: corresponding results

Screen-short for case #2:



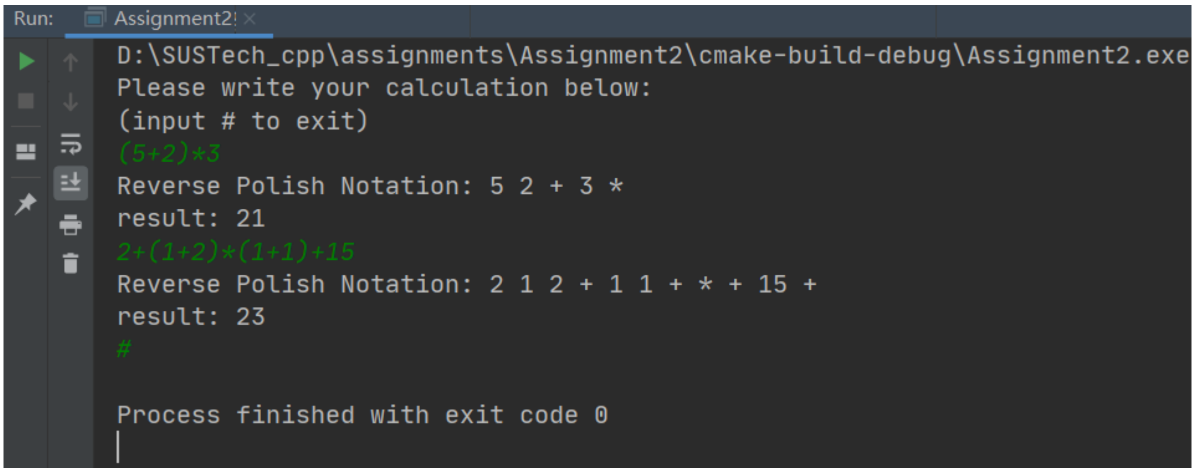
```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
1+1
Reverse Polish Notation: 1 1 +
result: 2
2*2
Reverse Polish Notation: 2 2 *
result: 4
248714+23495
Reverse Polish Notation: 248714 23495 +
result: 272209
sin(0)
Reverse Polish Notation: 0.000000
result: 0
ln(e)
Reverse Polish Notation: 1.000000
result: 1
#

Process finished with exit code 0
```

Test case #3(Use parentheses to enforce the priorities(可以正确处理优先级和括号)):

input: some calculations with parentheses

output: corresponding results

Screen-short for case #3:



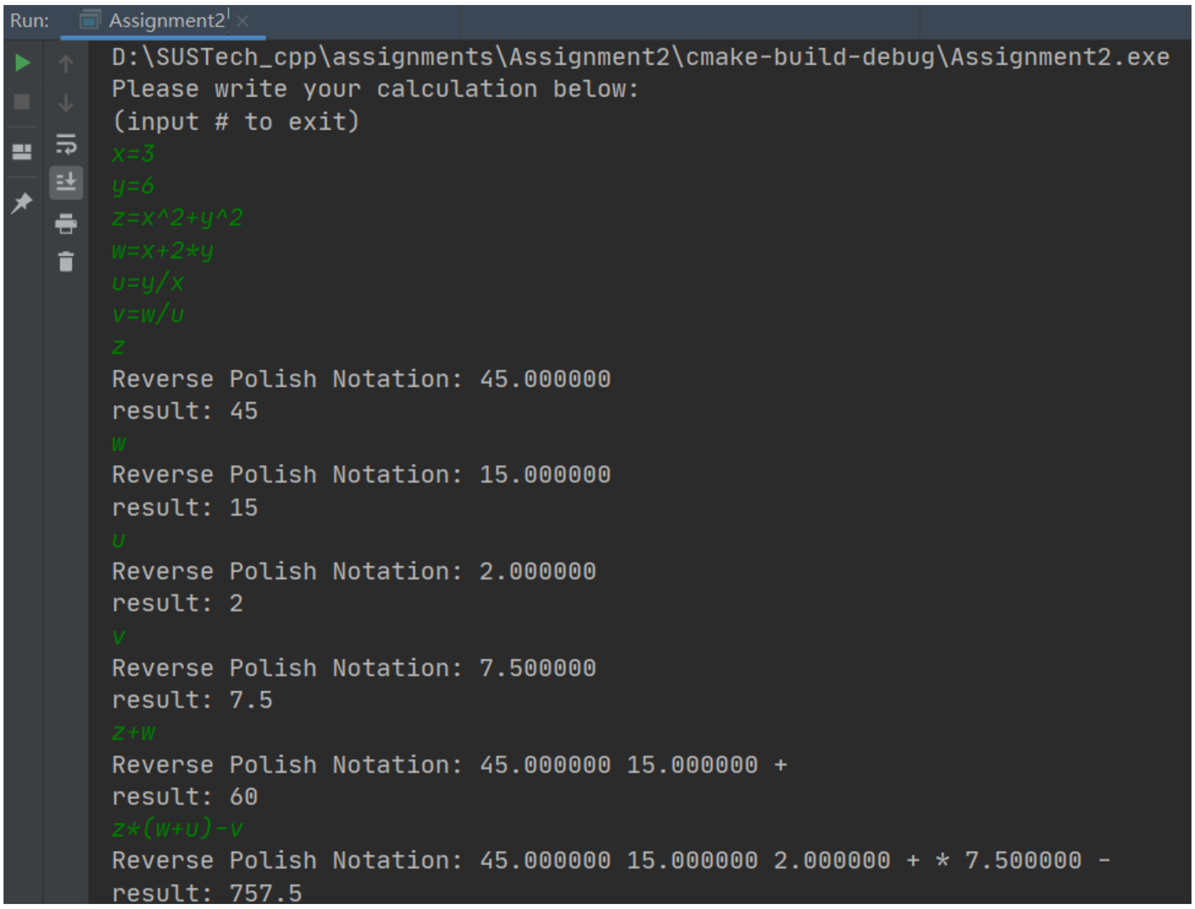
```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
(5+2)*3
Reverse Polish Notation: 5 2 + 3 *
result: 21
2+(1+2)*(1+1)+15
Reverse Polish Notation: 2 1 2 + 1 1 + * + 15 +
result: 23
#

Process finished with exit code 0
```

Test case #4(Variables can be defined(可以定义任何你想定义的函数，并让他参与计算)):

input: variables define

output: use defined variables to calculation

Screen-short for case #4:



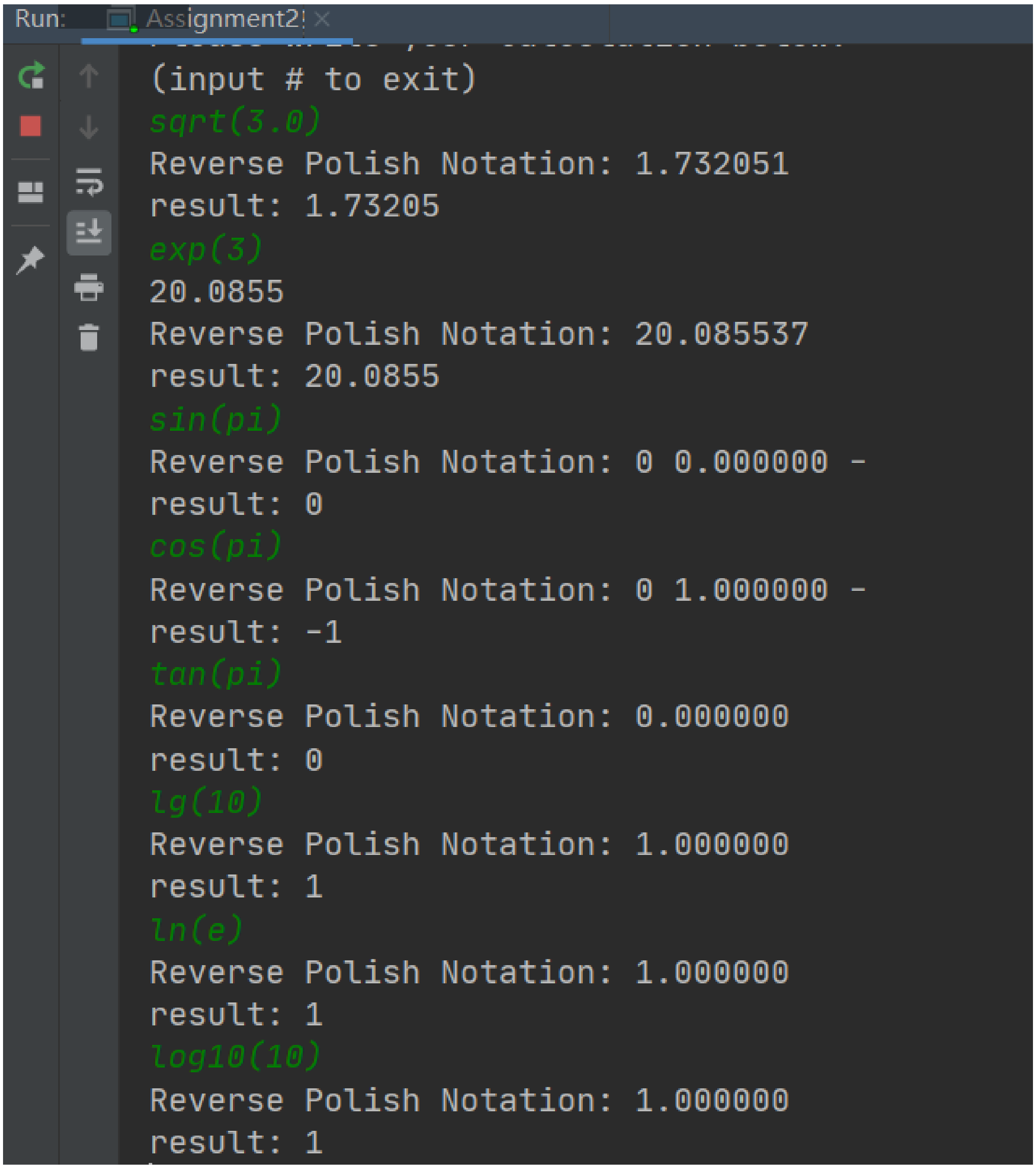
```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
x=3
y=6
z=x^2+y^2
w=x+2*y
u=y/x
v=w/u
z
Reverse Polish Notation: 45.000000
result: 45
w
Reverse Polish Notation: 15.000000
result: 15
u
Reverse Polish Notation: 2.000000
result: 2
v
Reverse Polish Notation: 7.500000
result: 7.5
z+w
Reverse Polish Notation: 45.000000 15.000000 +
result: 60
z*(w+u)-v
Reverse Polish Notation: 45.000000 15.000000 2.000000 + * 7.500000 -
result: 757.5
```

Test case #5(Some math functions can be supported(提供一些常用的数学函数，与数学中的符号规定相同，让你的计算更加便捷)):

input: some math functions

output: corresponding results

Screen-short for case #5:



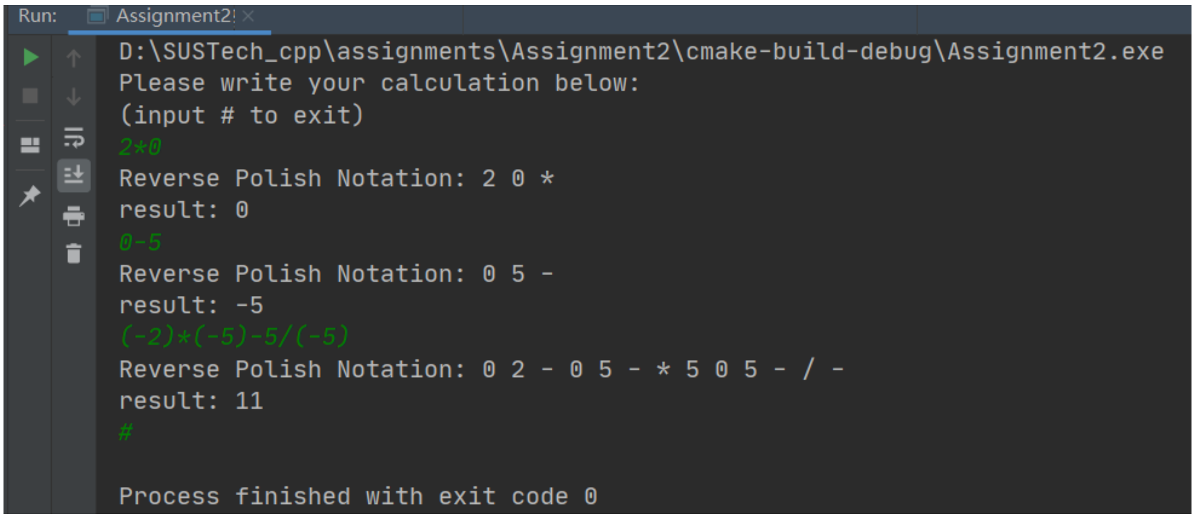
```
(input # to exit)
sqrt(3.0)
Reverse Polish Notation: 1.732051
result: 1.73205
exp(3)
20.0855
Reverse Polish Notation: 20.085537
result: 20.0855
sin(pi)
Reverse Polish Notation: 0 0.000000 -
result: 0
cos(pi)
Reverse Polish Notation: 0 1.000000 -
result: -1
tan(pi)
Reverse Polish Notation: 0.000000
result: 0
lg(10)
Reverse Polish Notation: 1.000000
result: 1
ln(e)
Reverse Polish Notation: 1.000000
result: 1
log10(10)
Reverse Polish Notation: 1.000000
result: 1
```

Test case #6(can deal with zero and negative numbers(它在遇到0的时候很够很聪明的应对，并能将它加入到负数的运算)):

input: zero and negative numbers

output: corresponding results

Screen-short for case #6:



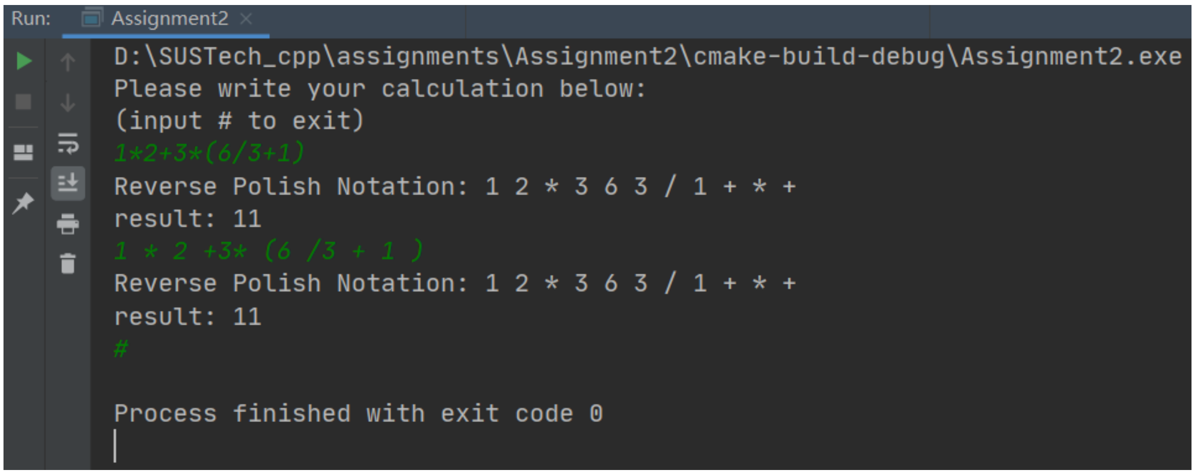
```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
2*0
Reverse Polish Notation: 2 0 *
result: 0
0-5
Reverse Polish Notation: 0 5 -
result: -5
(-2)*(-5)-5/(-5)
Reverse Polish Notation: 0 2 - 0 5 - * 5 0 5 - / -
result: 11
#

Process finished with exit code 0
```

Test case #7(can deal with random blank space(具有很好的兼容性，能够应对不同的输入格式)):

input: input with and without blank space

output: corresponding results

Screen-short for case #7:



```
D:\SUSTech_cpp\assignments\Assignment2\cmake-build-debug\Assignment2.exe
Please write your calculation below:
(input # to exit)
1*2+3*(6/3+1)
Reverse Polish Notation: 1 2 * 3 6 3 / 1 + * +
result: 11
1 * 2 +3* (6 /3 + 1 )
Reverse Polish Notation: 1 2 * 3 6 3 / 1 + * +
result: 11
#

Process finished with exit code 0
```

Part 3. Difficulties , Solutions and Core Algorithm

1.core algorithm:

using vector to imitate the data structure--stack, transform infix notation to Reverse Polish Notation which can be easily calculated from head to tail.

2.difficulties:

1)define variables:

use two vectors, one store the variable name and the other store the content, link them with the same index, the var name can not only be a letter, it can also be any word you want!

2)allow arbitrary blank space when input:

preprocess the input and erase the blank space

3)calculate with negative numbers:

preprocess the input and judge the result's negativity

4)allow basic math functions(sin, cos, tan, exp, sqrt, lg(以十为底), ln(以e为底), loga(b)(以a为底b的对数)， abs(绝对值))

we define a new function--scan to preprocess the input string, find the corresponding special functions and pre-calculate it into a numerical value. Especially, when we deal with loga(b), we use loga(b) = ln(b) / ln(a) to solve it!