Formula analyzer: Find the formula by parameters

Artur Eduardovich Sibgatullin
Kazan, Russian Federation

ridin9@gmail.com, skype: ridin7

Abstract

Let it be a formula, e.g.: $x + y^2 - z = r$. It is usually necessary to find a parameter's value by knowing others' ones. However, let's set another problem to find the formula itself, knowing only its parameters. The solution of such a problem we call reverse computing. For that we'll create an algorithm and accomplish it as a program code.

Index Terms

Reverse computing, formula analyzer, reverse Polish notation, reverse operation, reverse operation type.

Introduction

Let's take formula $x + y^2 - z = r$. Apparently, the formula may have the corresponding set 3,2,1, 6 : $3 + 2^2 - 1 = 6$, but this set also fits with formula $x + 2 * y - z$. It is obvious that we need a few sets of parameters for finding the formula with sufficient accuracy. Experience shows that it must be at least 5 such sets for 3 parameters for us to be able to separate accidentally coinciding formulas. For example: set $3 + 3^2 - 1 = 11$ does not already fit $3 + 2 * 3 - 1 = 8$. The formulas are expected to be found by simple item-by-item search of all possible parameters, limited by search parameters (code prototype in section Listing 1). Then we will have to set parameters and build an algorithm.

Methods

Presentation of the Formula

Let's use reverse Polish notation as it is understandable to a computer. In this our formula looks like $x, y^2, +, z, -$; i.e. first there are the parameters, and then the operations on them. The elements of the formula could be split into 4 types. Parameters: x; operations: $x + y$; properties of parameters and operations: $|x|, |x + y|$; constants: $x + 2$. But it is clear, that both properties and constants are either parameters, or operations, and it means we may cancel constants and partly cancel properties. For example: $|x + y| * 2$ it is one operation: $+|| * const$, and $|x| + 2$ it is a property: $|| + const$. Here a significant problem of standard computing is obvious: there could be an unlimited number of constant values in the formulas. It could be, for example, the number π. This problem will be considered in more detail later, and at this point let the constants take only the values of 1, 2, 3, 4. In such a way any formula for 3 parameters could be presented as:

TABLE 1: Formula

x	$\|x\|$	x_r	y	$\|y\|$	y_r	$+_1$	r_1	z	$\|z\|$	z_r	$+_2$	r_2
0	1	2	3	4	5	6	7	8	9	10	11	12

Where 0 element is parameter x, 1 is a property of the parameter x, 2 is a result of property x, 3 is parameter y, 4 is a property of the parameter y, 5 is a result of property y, 6 is the operation between results of properties x and y, 7 is a result of the first operation (6), 8 is parameter z, 9 is a property of the parameter z, 10 is a result of property z 11 is the operation between the result of the first operation and the result of property z, 12 is a result of the second operation (11)

However, is it really possible to present any formula through this method? Figuratively speaking, it is not exactly like that, because the sequence of parameters is important for us, or more exactly, the sequence of their interactions. For example, it is easy for us to express $x + y^2 - z$, but not $x/z + y$, because x first interacts with z, and not y. We don't know in advance the order in which our parameters interact. However, it is obvious that a set of 3 parameters could interact only in the following orders: 12+3+, 13+2+, 23+1+, where + is an operation between the entered parameters and computed at the previous step. (For the algorithm of finding the orders for a large number of parameters, see the section Discussion: Sequence with many parameters; the code for this algorithm is in the section Listing 2). It is also important for all operations to be commutative: $x + y$ equal to $y + x$, but $x - y$ is not equal to $y - x$. Because it is important that an order of parameters in an operation is of no importance, let us introduce for operations of ”minus type” $-$ operations of ”reverse minus type” $-^0$, making them in essence commutative: $x -^0 y = y - x$.

Algorithm

The reverse computing algorithm represents nested loops: to find a final result per 1 property, and per 2 properties, and per 1 operation, and per 3 properties, and per 2 operations.

Algorithm 1 Formula analyzer

```
Input: $x$, $y$, $z$, $result$
  for all Property $x$ do
    Result of $x$ = Property $x$
    for all Property $y$ do
      Result of $y$ = Property $y$
      for all First Operation do
        Result of First Operation = Operation between Results of $x$ and $y$
        for all Property $z$ do
          Result of $z$ = Property $z$
          for all Second Operation do
            Result of Second Operation = Operation between Result of $z$ and Result of First Operation
            if Result of Second Operation == $result$ then
              Check for another $x$, $y$, $z$, $result$
              if Check == true then
                return Property $x$, Property $y$, Property $z$, First Operation, Second Operation
              end if
            end if
          end for
        end for
      end for
    end for
  end for
```

First, you need to indicate all parameters and set operation and property numbers to 0. Then calculate property and operation result values up to the final result. Based on the values of previous properties and operations, search through all the numbers of the last operation to calculate values. After searching through all operation numbers, increase the last property number by 1, calculate the property, and recalculate it per operation number. Then for other properties and operations by analogy. At each final computation iteration, compare it to the search value. Upon a match, calculate the result for another parameter set by the formula found. On matching the search value, recheck for another parameter set. If a property or operation contains a $const$, searching through occurs also per the $const$ value. The $const$ is stored in a separate variable.

RESULTS AND SOLUTIONS

Similar Formulas

After we find a formula $x + y^2 - z = r$, we will be getting analogs of formulas, for example, $(x + 1 + y^2 - z) - 1 = r$. To check for an analog, it is necessary to write down separately ”a reference” (the first found formula), and while finding another one, to check the result for equality at similar sets of parameters (any), if they are equal at large number of parameters, then they are probably analogs, in an ideal case we’ll not be getting accidentally coinciding formulas that are not analogs.

Parameter input

The input parameters of a corresponding formula should be maximally random. For example, formulae $x + y^2 - z = r$ and $x * y + z - 3$ are apparently unequal, but with set 1,2,3 they equal 2, and with 2,3,4 = 7, 3,4,5 = 14, and so on ad inf. In each new set, parameters increase by 1. And if $x + y^2 - z$ and $x * y + z - 3$ are unequal, then $x + (x + 1)^2 - (x + 2)$ and $x * (x + 1) + (x + 2) - 3$ are equal. Another example: with parameters 3,5,1; 4,2,3; 5,7,3, formulae $x + y^2 - z$ and $x + y^2 + 3/z - 4$ are equal, because if $-z = 3/z - 4$, then $z^2 - 4z + 3 = 0, z_1 = 1, z_2 = 3$, then the formulae will be equal with such z values.

DISCUSSION. PROBLEMS AND SOLUTIONS.

Approximate Calculations

In reality, more than one force acts on an object, therefore there will be inaccuracies, interferences, and noises in observed phenomena. In future, the algorithm should be made even more complicated: to introduce an index of inaccuracy, with which the determined formula corresponds to the result; to save and to check formulas with the least index of inaccuracy.

Constants

The existence of constants increases the number of formulas' combinations to infinity. It is possible to increase gradually the range of possible constants values, including fractional. And to search by least error. Also, it is possible to try to create a complex algorithm in which all allowable constants will be calculated for each formula. E.g., for formula $x + const = r$, with the set $1 + const = 2$, $const$ is obviously equal to 1, and it could be calculated.

Sequence with many parameters

A multitude of parameters increases the sequence combination number. The algorithm itself may also change: set 1,2,3,4 may be calculated both as 12+34++ and 12+3+4+. Therefore, one has to search for different structures of 1 formula. A single algorithm is difficult to formulate, so let's consider set 1,2,3,4,5. We will have to enter the concept of valence, a parameter's ability to interact with others. Parameters interact pairwise, so it makes sense to divide them into pairs. $xy)nm)z$, the 1st pair parameters, interact only with each other as the 1st operation and then only as a pair (result). The 2nd pair parameters may come both as a pair and in isolation: 12+34++, 12+3+4+. The last parameter has no pair. In its turn, each pair may have all possible combinations, for example, the 1st pair: 12), 13), 14), 15), 23), 24), 25), 34), 35), 45). Parameter sequence does not matter, so we'll arrange them only in ascending order. Hence, the 1st pair 12) has possible 2nd pair combinations: 34), 35), 45). Then we search through all combinations for the 1st pair 12) whether the 2nd pair forms or not: 12+34++5+, 12+5+34++, 34+5+12++; 12+35++4+, 12+4+35++, 35+4+12++; 12+45++3+, 12+3+45++, 45+3+12++; 12+3+4+5+, 12+3+5+4+, 12+4+3+5+, 12+4+5+3+, 12+5+3+4+, 12+5+4+3+. Notably, the 1st pair may take on value 23) and the 2nd pair for it: 14), 15), 45). However, as pairs 14), 15) have already interacted with 23) at previous phases, they are not counted. (The code for this algorithm is in the section Listing 2)

CONCLUSIONS

After checking this algorithm against mathematical formulas, geometrical structures, and well-studied real world phenomena, we will be able to start analyzing other phenomena. It could allow us to discover patterns that had remained unnoticed, or happened to be too complicated for human perception. The main purpose of this algorithm is to transfer a part of research work to machines, so that they will be able to create new scientific knowledge.

HOW TO USE THE PROGRAM

The program is written in CodeBlocks on C. Sequentially input values x, y, z, r per 5 value groups. Wait for search completion after a full run lasts for 1 hour. The program will output the value set of the formula and constants, and the formula itself. The first 2 constants correspond to the x and y properties, the 3rd to the x and y operation, the 4th to the z property, and the 5th to the 2nd operation. After the program has found the formula, press enter to continue search and probably find formula analogs. If found, you will have to either change the algorithm or skip all analogs. Remember that output in C may differ from your computations. The program has challenges with power functions: $x^4 \neq (x^2)^2$ are often unequal as the double type is used that may have many symbols after the comma. The user had better input data directly to the code in order to be able to change the algorithm on their own. For example, if there occurs analogs irrelevant to computational accuracy, it makes sense to check them for another parameter set and add another checkup to the code. On finding an analog, there will be indicated above with what value set the results turned unequal. For example, enter for $x + y^2 - z = r$: 3,5,1,27; 4,2,3,5; 5,7,4,50; 4,12,7,141; 11,9,17,75;

REFERENCES

[1] Reverse Polish notation:
`https://en.wikipedia.org/wiki/Reverse_Polish_notation`

[2] C mathematical functions:
`https://en.wikipedia.org/wiki/C_mathematical_functions`

[3] Stanislaw Lem, Sum of Technology 1964 (1967 - second edition). Chapter 7: Creation of the Worlds (”grow” new information).

Listing 1: Formula analyzer

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include <math.h>
#define MAX 51

double formula[15],formula2[15], formula3[15], formula4[15],formula5[15], check[27][17], standart
double constant[8], stconstant[8];
int indx, indy, constf, flagc, flagc1, flagc2, IsStand = 0, IsCheck = 0;

int main()
{
int i,j,n,l,m;

constant[0] = 1;
constant[1] = 1;
constant[2] = 1;
constant[3] = 1;
constant[4] = 1;

Scan();
/*
//x+y^2-z = answer 3 5 1 = 27, 4 2 3 = 5

formula[0] = 3;
formula[3] = 5;
formula[8] = 1;
formula[13] = 27;

formula2[0] = 4;
formula2[3] = 2;
formula2[8] = 3;
formula2[13] = 5;

formula3[0] = 5;
formula3[3] = 7;
formula3[8] = 4;
formula3[13] = 50;

formula4[0] = 4;
formula4[3] = 12;
formula4[8] = 7;
formula4[13] = 141;

formula5[0] = 11;
formula5[3] = 9;
formula5[8] = 17;
formula5[13] = 75;
*/
for(i=0;i!=17;i++){
    if(constant[0] != 5 && constant[0] != 1){formula[1]--;}
    if(constant[0] == 5) constant[0] = 1;
    Property(0,i,0);
    if(constant[0] != 5 && constant[0] != 1){i--;}
    formula[4] = 0;
    formula[1]++;

    for(j=0;j!=17;j++){
        if(constant[1] != 5 && constant[1] != 1){formula[4]--;}
        if(constant[1] == 5) constant[1] = 1;
        Property(3,j,1);
        if(constant[1] != 5 && constant[1] != 1){j--;}
        formula[6] = 0;
        formula[4]++;

        for(n=0;n!=144;n++){
            if(constant[2] != 5 && constant[2] != 1){formula[6]--;}
            if(constant[2] == 5) constant[2] = 1;
            Operation(2,5,n,2);
            if(constant[2] != 5 && constant[2] != 1){n--;}
            formula[9] = 0;
            formula[6]++;

            for(l=0;l!=17;l++){
                if(constant[3] != 5 && constant[3] != 1){formula[9]--;}
                if(constant[3] == 5) constant[3] = 1;
                Property(8,l,3);
                if(constant[3] != 5 && constant[3] != 1){l--;}
                formula[11] = 0;
                formula[9]++;

                for(m=0;m!=144;m++){
                    if(constant[4] != 5 && constant[4] != 1){formula[11]--;}
                    if(constant[4] == 5)constant[4] = 1;
                    Operation(7,10,m,4);
                    if(constant[4] != 5 && constant[4] != 1){m--;}
                    formula[11]++;
                    if(fabs(formula[12] - formula[13]) < 0.0001){Another();}
                }
            }
        }
    }
}
printf("Complete");
getch();
}

void Property(int indf,int casep,int cons)
{
    switch ( casep ) {
        case 0:formula[indf + 2] = formula[indf];break;
        case 1:formula[indf + 2] = fabs(formula[indf]);break;
        case 2:formula[indf + 2] = sin(formula[indf]);break;
        case 3:formula[indf + 2] = cos(formula[indf]);break;
        case 4:formula[indf + 2] = tan(formula[indf]);break;
        case 5:formula[indf + 2] = log(formula[indf]);break;
        case 6:formula[indf + 2] = log10(formula[indf]);break;
        case 7:formula[indf + 2] = formula[indf] + constant[cons];constant[cons]++;break;
        case 8:formula[indf + 2] = formula[indf] * constant[cons];constant[cons]++;break;
        case 9:formula[indf + 2] = formula[indf] / constant[cons];constant[cons]++;break;
        case 10:formula[indf + 2] = constant[cons] / formula[indf];constant[cons]++;break;
        case 11:formula[indf + 2] = formula[indf] - constant[cons];constant[cons]++;break;
        case 12:formula[indf + 2] = constant[cons] - formula[indf];constant[cons]++;break;
        case 13:formula[indf + 2] = formula[indf] * constant[cons] * -1;constant[cons]++;break;
        case 14:formula[indf + 2] = pow(formula[indf],constant[cons]);constant[cons]++;break;
        case 15:formula[indf + 2] = pow(constant[cons],formula[indf]);constant[cons]++;break;
        case 16:formula[indf + 2] = pow(formula[indf],1 / constant[cons]);constant[cons]++;break;
    }
}

void Operation(int indx, int indy, int caseo,int cons)
{
if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17); flag
else flagc = 0;
switch ( caseo ) {
        case 0:formula[indy + 2] = formula[indx] + formula[indy];break;
        case 1:formula[indy + 2] = formula[indx] * formula[indy];break;
        case 2:formula[indy + 2] = formula[indx] - formula[indy];break;
        case 3:formula[indy + 2] = formula[indy] - formula[indx];break;
        case 4:formula[indy + 2] = formula[indx] / formula[indy];break;
        case 5:formula[indy + 2] = formula[indy] / formula[indx];break;
        case 6:formula[indy + 2] = pow(formula[indx],formula[indy]);break;
        case 7:formula[indy + 2] = pow(formula[indy],formula[indx]);break;
    }

if(flagc == 1)
switch ( constf ) {
        case 0:formula[indy + 2] = formula[indy + 2];break;
        case 1:formula[indy + 2] = fabs(formula[indy + 2]);break;
        case 2:formula[indy + 2] = sin(formula[indy + 2]);break;
        case 3:formula[indy + 2] = cos(formula[indy + 2]);break;
        case 4:formula[indy + 2] = tan(formula[indy + 2]);break;
        case 5:formula[indy + 2] = log(formula[indy + 2]);break;
        case 6:formula[indy + 2] = log10(formula[indy + 2]);break;
        case 7:formula[indy + 2] = formula[indy + 2] + constant[cons];constant[cons]++;break;
        case 8:formula[indy + 2] = formula[indy + 2] * constant[cons];constant[cons]++;break;
        case 9:formula[indy + 2] = formula[indy + 2] / constant[cons];constant[cons]++;break;
        case 10:formula[indy + 2] = constant[cons] / formula[indy + 2];constant[cons]++;break;
        case 11:formula[indy + 2] = formula[indy + 2] - constant[cons];constant[cons]++;break;
        case 12:formula[indy + 2] = constant[cons] - formula[indy + 2];constant[cons]++;break;
        case 13:formula[indy + 2] = formula[indy + 2] * constant[cons] * -1;constant[cons]++;break
        case 14:formula[indy + 2] = pow(formula[indy + 2],constant[cons]);constant[cons]++;break;
        case 15:formula[indy + 2] = pow(constant[cons],formula[indy + 2]);constant[cons]++;break;
        case 16:formula[indy + 2] = pow(formula[indy + 2],1 / constant[cons]);constant[cons]++;bre
    }
}

void Another() // Check for another parameters
{
int i, caseo, caseo2;

switch ( (int)(formula[1] - 1) ) {
        case 0:formula2[2] = formula2[0];break;
        case 1:formula2[2] = fabs(formula2[0]);break;
        case 2:formula2[2] = sin(formula2[0]);break;
        case 3:formula2[2] = cos(formula2[0]);break;
        case 4:formula2[2] = tan(formula2[0]);break;
        case 5:formula2[2] = log(formula2[0]);break;
        case 6:formula2[2] = log10(formula2[0]);break;
        case 7:formula2[2] = formula2[0] + (constant[0] - 1);break;
        case 8:formula2[2] = formula2[0] * (constant[0] - 1);break;
        case 9:formula2[2] = formula2[0] / (constant[0] - 1);break;
        case 10:formula2[2] = (constant[0] - 1) / formula2[0];break;
        case 11:formula2[2] = formula2[0] - (constant[0] - 1);break;
        case 12:formula2[2] = (constant[0] - 1) - formula2[0];break;
        case 13:formula2[2] = formula2[0] * (constant[0] - 1) * -1;break;
        case 14:formula2[2] = pow(formula2[0],(constant[0] - 1));break;
        case 15:formula2[2] = pow((constant[0] - 1),formula2[0]);break;
        case 16:formula2[2] = pow(formula2[0],1 / (constant[0] - 1));break;
    }

switch ( (int)(formula[4] - 1) ) {
        case 0:formula2[5] = formula2[3];break;
        case 1:formula2[5] = fabs(formula2[3]);break;
        case 2:formula2[5] = sin(formula2[3]);break;
        case 3:formula2[5] = cos(formula2[3]);break;
        case 4:formula2[5] = tan(formula2[3]);break;
        case 5:formula2[5] = log(formula2[3]);break;
        case 6:formula2[5] = log10(formula2[3]);break;
        case 7:formula2[5] = formula2[3] + (constant[1] - 1);break;
        case 8:formula2[5] = formula2[3] * (constant[1] - 1);break;
        case 9:formula2[5] = formula2[3] / (constant[1] - 1);break;
        case 10:formula2[5] = (constant[1] - 1) / formula2[3];break;
        case 11:formula2[5] = formula2[3] - (constant[1] - 1);break;
        case 12:formula2[5] = (constant[1] - 1) - formula2[3];break;
        case 13:formula2[5] = formula2[3] * (constant[1] - 1) * -1;break;
        case 14:formula2[5] = pow(formula2[3],(constant[1] - 1));break;
        case 15:formula2[5] = pow((constant[1] - 1),formula2[3]);break;
        case 16:formula2[5] = pow(formula2[3],1 / (constant[1] - 1));break;
    }

    switch ( (int)(formula[9] - 1) ) {
        case 0:formula2[10] = formula2[8];break;
        case 1:formula2[10] = fabs(formula2[8]);break;
        case 2:formula2[10] = sin(formula2[8]);break;
        case 3:formula2[10] = cos(formula2[8]);break;
        case 4:formula2[10] = tan(formula2[8]);break;
        case 5:formula2[10] = log(formula2[8]);break;
        case 6:formula2[10] = log10(formula2[8]);break;
        case 7:formula2[10] = formula2[8] + (constant[3] - 1);break;
        case 8:formula2[10] = formula2[8] * (constant[3] - 1);break;
        case 9:formula2[10] = formula2[8] / (constant[3] - 1);break;
        case 10:formula2[10] = (constant[3] - 1) / formula2[8];break;
        case 11:formula2[10] = formula2[8] - (constant[3] - 1);break;
        case 12:formula2[10] = (constant[3] - 1) - formula2[8];break;
        case 13:formula2[10] = formula2[8] * (constant[3] - 1) * -1;break;
        case 14:formula2[10] = pow(formula2[8],(constant[3] - 1));break;
        case 15:formula2[10] = pow((constant[3] - 1),formula2[8]);break;
        case 16:formula2[10] = pow(formula2[8],1 / (constant[3] - 1));break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:formula2[7] = formula2[2] + formula2[5];break;
        case 1:formula2[7] = formula2[2] * formula2[5];break;
        case 2:formula2[7] = formula2[2] - formula2[5];break;
        case 3:formula2[7] = formula2[5] - formula2[2];break;
        case 4:formula2[7] = formula2[2] / formula2[5];break;
        case 5:formula2[7] = formula2[5] / formula2[2];break;
        case 6:formula2[7] = pow(formula2[2],formula2[5]);break;
        case 7:formula2[7] = pow(formula2[5],formula2[2]);break;
    }

if(flagc1 == 1)
```

```
switch ( constf ) {
        case 0:formula2[7] = formula2[7];break;
        case 1:formula2[7] = fabs(formula2[7]);break;
        case 2:formula2[7] = sin(formula2[7]);break;
        case 3:formula2[7] = cos(formula2[7]);break;
        case 4:formula2[7] = tan(formula2[7]);break;
        case 5:formula2[7] = log(formula2[7]);break;
        case 6:formula2[7] = log10(formula2[7]);break;
        case 7:formula2[7] = formula2[7] + (constant[2] - 1);break;
        case 8:formula2[7] = formula2[7] * (constant[2] - 1);break;
        case 9:formula2[7] = formula2[7] / (constant[2] - 1);break;
        case 10:formula2[7] = (constant[2] - 1) / formula2[7];break;
        case 11:formula2[7] = formula2[7] - (constant[2] - 1);break;
        case 12:formula2[7] = (constant[2] - 1) - formula2[7];break;
        case 13:formula2[7] = formula2[7] * (constant[2] - 1) * -1;break;
        case 14:formula2[7] = pow(formula2[7],(constant[2] - 1));break;
        case 15:formula2[7] = pow((constant[2] - 1),formula2[7]);break;
        case 16:formula2[7] = pow(formula2[7],1 / (constant[2] - 1));break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:formula2[12] = formula2[7] + formula2[10];break;
        case 1:formula2[12] = formula2[7] * formula2[10];break;
        case 2:formula2[12] = formula2[7] - formula2[10];break;
        case 3:formula2[12] = formula2[10] - formula2[7];break;
        case 4:formula2[12] = formula2[7] / formula2[10];break;
        case 5:formula2[12] = formula2[10] / formula2[7];break;
        case 6:formula2[12] = pow(formula2[7],formula2[10]);break;
        case 7:formula2[12] = pow(formula2[10],formula2[7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:formula2[12] = formula2[12];break;
        case 1:formula2[12] = fabs(formula2[12]);break;
        case 2:formula2[12] = sin(formula2[12]);break;
        case 3:formula2[12] = cos(formula2[12]);break;
        case 4:formula2[12] = tan(formula2[12]);break;
        case 5:formula2[12] = log(formula2[12]);break;
        case 6:formula2[12] = log10(formula2[12]);break;
        case 7:formula2[12] = formula2[12] + (constant[4] - 1);break;
        case 8:formula2[12] = formula2[12] * (constant[4] - 1);break;
        case 9:formula2[12] = formula2[12] / (constant[4] - 1);break;
        case 10:formula2[12] = (constant[4] - 1) / formula2[12];break;
        case 11:formula2[12] = formula2[12] - (constant[4] - 1);break;
        case 12:formula2[12] = (constant[4] - 1) - formula2[12];break;
        case 13:formula2[12] = formula2[12] * (constant[4] - 1) * -1;break;
        case 14:formula2[12] = pow(formula2[12],(constant[4] - 1));break;
        case 15:formula2[12] = pow((constant[4] - 1),formula2[12]);break;
        case 16:formula2[12] = pow(formula2[12],1 / (constant[4] - 1));break;
    }

    if(formula[1]==15 && formula[4]==1 && formula[6]==5 && formula[9]==1 && formula[11]==7 && con
{
    Print2();
}
if(fabs(formula2[12] - formula2[13]) < 0.0001){
    Another2();
}
}
```

```
void Another2()
{
int i, caseo, caseo2;

switch ( (int)(formula[1] − 1) ) {
        case 0:formula3[2] = formula3[0];break;
        case 1:formula3[2] = fabs(formula3[0]);break;
        case 2:formula3[2] = sin(formula3[0]);break;
        case 3:formula3[2] = cos(formula3[0]);break;
        case 4:formula3[2] = tan(formula3[0]);break;
        case 5:formula3[2] = log(formula3[0]);break;
        case 6:formula3[2] = log10(formula3[0]);break;
        case 7:formula3[2] = formula3[0] + (constant[0] − 1);break;
        case 8:formula3[2] = formula3[0] ∗ (constant[0] − 1);break;
        case 9:formula3[2] = formula3[0] / (constant[0] − 1);break;
        case 10:formula3[2] = (constant[0] − 1) / formula3[0];break;
        case 11:formula3[2] = formula3[0] − (constant[0] − 1);break;
        case 12:formula3[2] = (constant[0] − 1) − formula3[0];break;
        case 13:formula3[2] = formula3[0] ∗ (constant[0] − 1) ∗ −1;break;
        case 14:formula3[2] = pow(formula3[0],(constant[0] − 1));break;
        case 15:formula3[2] = pow((constant[0] − 1),formula3[0]);break;
        case 16:formula3[2] = pow(formula3[0],1 / (constant[0] − 1));break;
    }

switch ( (int)(formula[4] − 1) ) {
        case 0:formula3[5] = formula3[3];break;
        case 1:formula3[5] = fabs(formula3[3]);break;
        case 2:formula3[5] = sin(formula3[3]);break;
        case 3:formula3[5] = cos(formula3[3]);break;
        case 4:formula3[5] = tan(formula3[3]);break;
        case 5:formula3[5] = log(formula3[3]);break;
        case 6:formula3[5] = log10(formula3[3]);break;
        case 7:formula3[5] = formula3[3] + (constant[1] − 1);break;
        case 8:formula3[5] = formula3[3] ∗ (constant[1] − 1);break;
        case 9:formula3[5] = formula3[3] / (constant[1] − 1);break;
        case 10:formula3[5] = (constant[1] − 1) / formula3[3];break;
        case 11:formula3[5] = formula3[3] − (constant[1] − 1);break;
        case 12:formula3[5] = (constant[1] − 1) − formula3[3];break;
        case 13:formula3[5] = formula3[3] ∗ (constant[1] − 1) ∗ −1;break;
        case 14:formula3[5] = pow(formula3[3],(constant[1] − 1));break;
        case 15:formula3[5] = pow((constant[1] − 1),formula3[3]);break;
        case 16:formula3[5] = pow(formula3[3],1 / (constant[1] − 1));break;
    }

    switch ( (int)(formula[9] − 1) ) {
        case 0:formula3[10] = formula3[8];break;
        case 1:formula3[10] = fabs(formula3[8]);break;
        case 2:formula3[10] = sin(formula3[8]);break;
        case 3:formula3[10] = cos(formula3[8]);break;
        case 4:formula3[10] = tan(formula3[8]);break;
        case 5:formula3[10] = log(formula3[8]);break;
        case 6:formula3[10] = log10(formula3[8]);break;
        case 7:formula3[10] = formula3[8] + (constant[3] − 1);break;
        case 8:formula3[10] = formula3[8] ∗ (constant[3] − 1);break;
        case 9:formula3[10] = formula3[8] / (constant[3] − 1);break;
        case 10:formula3[10] = (constant[3] − 1) / formula3[8];break;
        case 11:formula3[10] = formula3[8] − (constant[3] − 1);break;
        case 12:formula3[10] = (constant[3] − 1) − formula3[8];break;
        case 13:formula3[10] = formula3[8] ∗ (constant[3] − 1) ∗ −1;break;
        case 14:formula3[10] = pow(formula3[8],(constant[3] − 1));break;
        case 15:formula3[10] = pow((constant[3] - 1),formula3[8]);break;
        case 16:formula3[10] = pow(formula3[8],1 / (constant[3] - 1));break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:formula3[7] = formula3[2] + formula3[5];break;
        case 1:formula3[7] = formula3[2] * formula3[5];break;
        case 2:formula3[7] = formula3[2] - formula3[5];break;
        case 3:formula3[7] = formula3[5] - formula3[2];break;
        case 4:formula3[7] = formula3[2] / formula3[5];break;
        case 5:formula3[7] = formula3[5] / formula3[2];break;
        case 6:formula3[7] = pow(formula3[2],formula3[5]);break;
        case 7:formula3[7] = pow(formula3[5],formula3[2]);break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:formula3[7] = formula3[7];break;
        case 1:formula3[7] = fabs(formula3[7]);break;
        case 2:formula3[7] = sin(formula3[7]);break;
        case 3:formula3[7] = cos(formula3[7]);break;
        case 4:formula3[7] = tan(formula3[7]);break;
        case 5:formula3[7] = log(formula3[7]);break;
        case 6:formula3[7] = log10(formula3[7]);break;
        case 7:formula3[7] = formula3[7] + (constant[2] - 1);break;
        case 8:formula3[7] = formula3[7] * (constant[2] - 1);break;
        case 9:formula3[7] = formula3[7] / (constant[2] - 1);break;
        case 10:formula3[7] = (constant[2] - 1) / formula3[7];break;
        case 11:formula3[7] = formula3[7] - (constant[2] - 1);break;
        case 12:formula3[7] = (constant[2] - 1) - formula3[7];break;
        case 13:formula3[7] = formula3[7] * (constant[2] - 1) * -1;break;
        case 14:formula3[7] = pow(formula3[7],(constant[2] - 1));break;
        case 15:formula3[7] = pow((constant[2] - 1),formula3[7]);break;
        case 16:formula3[7] = pow(formula3[7],1 / (constant[2] - 1));break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:formula3[12] = formula3[7] + formula3[10];break;
        case 1:formula3[12] = formula3[7] * formula3[10];break;
        case 2:formula3[12] = formula3[7] - formula3[10];break;
        case 3:formula3[12] = formula3[10] - formula3[7];break;
        case 4:formula3[12] = formula3[7] / formula3[10];break;
        case 5:formula3[12] = formula3[10] / formula3[7];break;
        case 6:formula3[12] = pow(formula3[7],formula3[10]);break;
        case 7:formula3[12] = pow(formula3[10],formula3[7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:formula3[12] = formula3[12];break;
        case 1:formula3[12] = fabs(formula3[12]);break;
        case 2:formula3[12] = sin(formula3[12]);break;
        case 3:formula3[12] = cos(formula3[12]);break;
        case 4:formula3[12] = tan(formula3[12]);break;
        case 5:formula3[12] = log(formula3[12]);break;
        case 6:formula3[12] = log10(formula3[12]);break;
        case 7:formula3[12] = formula3[12] + (constant[4] - 1);break;
        case 8:formula3[12] = formula3[12] * (constant[4] - 1);break;
        case 9:formula3[12] = formula3[12] / (constant[4] - 1);break;
        case 10:formula3[12] = (constant[4] - 1) / formula3[12];break;
        case 11:formula3[12] = formula3[12] - (constant[4] - 1);break;
        case 12:formula3[12] = (constant[4] - 1) - formula3[12];break;
        case 13:formula3[12] = formula3[12] * (constant[4] - 1) * -1;break;
        case 14:formula3[12] = pow(formula3[12],(constant[4] - 1));break;
        case 15:formula3[12] = pow((constant[4] - 1),formula3[12]);break;
        case 16:formula3[12] = pow(formula3[12],1 / (constant[4] - 1));break;
    }
if(formula[1]==15 && formula[4]==1 && formula[6]==5 && formula[9]==1 && formula[11]==7 && constant
{
    Print3();
}
if(fabs(formula3[12] - formula3[13]) < 0.0001){
Another3();
}
}

void Another3()
{
int i, caseo, caseo2;

switch ( (int)(formula[1] - 1) ) {
        case 0:formula4[2] = formula4[0];break;
        case 1:formula4[2] = fabs(formula4[0]);break;
        case 2:formula4[2] = sin(formula4[0]);break;
        case 3:formula4[2] = cos(formula4[0]);break;
        case 4:formula4[2] = tan(formula4[0]);break;
        case 5:formula4[2] = log(formula4[0]);break;
        case 6:formula4[2] = log10(formula4[0]);break;
        case 7:formula4[2] = formula4[0] + (constant[0] - 1);break;
        case 8:formula4[2] = formula4[0] * (constant[0] - 1);break;
        case 9:formula4[2] = formula4[0] / (constant[0] - 1);break;
        case 10:formula4[2] = (constant[0] - 1) / formula4[0];break;
        case 11:formula4[2] = formula4[0] - (constant[0] - 1);break;
        case 12:formula4[2] = (constant[0] - 1) - formula4[0];break;
        case 13:formula4[2] = formula4[0] * (constant[0] - 1) * -1;break;
        case 14:formula4[2] = pow(formula4[0],(constant[0] - 1));break;
        case 15:formula4[2] = pow((constant[0] - 1),formula4[0]);break;
        case 16:formula4[2] = pow(formula4[0],1 / (constant[0] - 1));break;
    }

switch ( (int)(formula[4] - 1) ) {
        case 0:formula4[5] = formula4[3];break;
        case 1:formula4[5] = fabs(formula4[3]);break;
        case 2:formula4[5] = sin(formula4[3]);break;
        case 3:formula4[5] = cos(formula4[3]);break;
        case 4:formula4[5] = tan(formula4[3]);break;
        case 5:formula4[5] = log(formula4[3]);break;
        case 6:formula4[5] = log10(formula4[3]);break;
        case 7:formula4[5] = formula4[3] + (constant[1] - 1);break;
        case 8:formula4[5] = formula4[3] * (constant[1] - 1);break;
        case 9:formula4[5] = formula4[3] / (constant[1] - 1);break;
        case 10:formula4[5] = (constant[1] - 1) / formula4[3];break;
        case 11:formula4[5] = formula4[3] - (constant[1] - 1);break;
        case 12:formula4[5] = (constant[1] - 1) - formula4[3];break;
        case 13:formula4[5] = formula4[3] * (constant[1] - 1) * -1;break;
        case 14:formula4[5] = pow(formula4[3],(constant[1] - 1));break;
        case 15:formula4[5] = pow((constant[1] - 1),formula4[3]);break;
        case 16:formula4[5] = pow(formula4[3],1 / (constant[1] - 1));break;
    }

    switch ( (int)(formula[9] - 1) ) {
        case 0:formula4[10] = formula4[8];break;
        case 1:formula4[10] = fabs(formula4[8]);break;
        case 2:formula4[10] = sin(formula4[8]);break;
        case 3:formula4[10] = cos(formula4[8]);break;
        case 4:formula4[10] = tan(formula4[8]);break;
        case 5:formula4[10] = log(formula4[8]);break;
        case 6:formula4[10] = log10(formula4[8]);break;
        case 7:formula4[10] = formula4[8] + (constant[3] - 1);break;
        case 8:formula4[10] = formula4[8] * (constant[3] - 1);break;
        case 9:formula4[10] = formula4[8] / (constant[3] - 1);break;
        case 10:formula4[10] = (constant[3] - 1) / formula4[8];break;
        case 11:formula4[10] = formula4[8] - (constant[3] - 1);break;
        case 12:formula4[10] = (constant[3] - 1) - formula4[8];break;
        case 13:formula4[10] = formula4[8] * (constant[3] - 1) * -1;break;
        case 14:formula4[10] = pow(formula4[8],(constant[3] - 1));break;
        case 15:formula4[10] = pow((constant[3] - 1),formula4[8]);break;
        case 16:formula4[10] = pow(formula4[8],1 / (constant[3] - 1));break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:formula4[7] = formula4[2] + formula4[5];break;
        case 1:formula4[7] = formula4[2] * formula4[5];break;
        case 2:formula4[7] = formula4[2] - formula4[5];break;
        case 3:formula4[7] = formula4[5] - formula4[2];break;
        case 4:formula4[7] = formula4[2] / formula4[5];break;
        case 5:formula4[7] = formula4[5] / formula4[2];break;
        case 6:formula4[7] = pow(formula4[2],formula4[5]);break;
        case 7:formula4[7] = pow(formula4[5],formula4[2]);break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:formula4[7] = formula4[7];break;
        case 1:formula4[7] = fabs(formula4[7]);break;
        case 2:formula4[7] = sin(formula4[7]);break;
        case 3:formula4[7] = cos(formula4[7]);break;
        case 4:formula4[7] = tan(formula4[7]);break;
        case 5:formula4[7] = log(formula4[7]);break;
        case 6:formula4[7] = log10(formula4[7]);break;
        case 7:formula4[7] = formula4[7] + (constant[2] - 1);break;
        case 8:formula4[7] = formula4[7] * (constant[2] - 1);break;
        case 9:formula4[7] = formula4[7] / (constant[2] - 1);break;
        case 10:formula4[7] = (constant[2] - 1) / formula4[7];break;
        case 11:formula4[7] = formula4[7] - (constant[2] - 1);break;
        case 12:formula4[7] = (constant[2] - 1) - formula4[7];break;
        case 13:formula4[7] = formula4[7] * (constant[2] - 1) * -1;break;
        case 14:formula4[7] = pow(formula4[7],(constant[2] - 1));break;
        case 15:formula4[7] = pow((constant[2] - 1),formula4[7]);break;
        case 16:formula4[7] = pow(formula4[7],1 / (constant[2] - 1));break;
    }
```

```
if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:formula4[12] = formula4[7] + formula4[10];break;
        case 1:formula4[12] = formula4[7] * formula4[10];break;
        case 2:formula4[12] = formula4[7] - formula4[10];break;
        case 3:formula4[12] = formula4[10] - formula4[7];break;
        case 4:formula4[12] = formula4[7] / formula4[10];break;
        case 5:formula4[12] = formula4[10] / formula4[7];break;
        case 6:formula4[12] = pow(formula4[7],formula4[10]);break;
        case 7:formula4[12] = pow(formula4[10],formula4[7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:formula4[12] = formula4[12];break;
        case 1:formula4[12] = fabs(formula4[12]);break;
        case 2:formula4[12] = sin(formula4[12]);break;
        case 3:formula4[12] = cos(formula4[12]);break;
        case 4:formula4[12] = tan(formula4[12]);break;
        case 5:formula4[12] = log(formula4[12]);break;
        case 6:formula4[12] = log10(formula4[12]);break;
        case 7:formula4[12] = formula4[12] + (constant[4] - 1);break;
        case 8:formula4[12] = formula4[12] * (constant[4] - 1);break;
        case 9:formula4[12] = formula4[12] / (constant[4] - 1);break;
        case 10:formula4[12] = (constant[4] - 1) / formula4[12];break;
        case 11:formula4[12] = formula4[12] - (constant[4] - 1);break;
        case 12:formula4[12] = (constant[4] - 1) - formula4[12];break;
        case 13:formula4[12] = formula4[12] * (constant[4] - 1) * -1;break;
        case 14:formula4[12] = pow(formula4[12],(constant[4] - 1));break;
        case 15:formula4[12] = pow((constant[4] - 1),formula4[12]);break;
        case 16:formula4[12] = pow(formula4[12],1 / (constant[4] - 1));break;
    }

if(fabs(formula4[12] - formula4[13]) < 0.0001){
Another4();
}
}

void Another4()
{
int i, caseo, caseo2;

switch ( (int)(formula[1] - 1) ) {
        case 0:formula5[2] = formula5[0];break;
        case 1:formula5[2] = fabs(formula5[0]);break;
        case 2:formula5[2] = sin(formula5[0]);break;
        case 3:formula5[2] = cos(formula5[0]);break;
        case 4:formula5[2] = tan(formula5[0]);break;
        case 5:formula5[2] = log(formula5[0]);break;
        case 6:formula5[2] = log10(formula5[0]);break;
        case 7:formula5[2] = formula5[0] + (constant[0] - 1);break;
        case 8:formula5[2] = formula5[0] * (constant[0] - 1);break;
        case 9:formula5[2] = formula5[0] / (constant[0] - 1);break;
        case 10:formula5[2] = (constant[0] - 1) / formula5[0];break;
        case 11:formula5[2] = formula5[0] - (constant[0] - 1);break;
        case 12:formula5[2] = (constant[0] - 1) - formula5[0];break;
        case 13:formula5[2] = formula5[0] * (constant[0] - 1) * -1;break;
        case 14:formula5[2] = pow(formula5[0],(constant[0] - 1));break;
        case 15:formula5[2] = pow((constant[0] - 1),formula5[0]);break;
        case 16:formula5[2] = pow(formula5[0],1 / (constant[0] - 1));break;
    }
```

```
switch ( (int)(formula[4] - 1) ) {
        case 0:formula5[5] = formula5[3];break;
        case 1:formula5[5] = fabs(formula5[3]);break;
        case 2:formula5[5] = sin(formula5[3]);break;
        case 3:formula5[5] = cos(formula5[3]);break;
        case 4:formula5[5] = tan(formula5[3]);break;
        case 5:formula5[5] = log(formula5[3]);break;
        case 6:formula5[5] = log10(formula5[3]);break;
        case 7:formula5[5] = formula5[3] + (constant[1] - 1);break;
        case 8:formula5[5] = formula5[3] * (constant[1] - 1);break;
        case 9:formula5[5] = formula5[3] / (constant[1] - 1);break;
        case 10:formula5[5] = (constant[1] - 1) / formula5[3];break;
        case 11:formula5[5] = formula5[3] - (constant[1] - 1);break;
        case 12:formula5[5] = (constant[1] - 1) - formula5[3];break;
        case 13:formula5[5] = formula5[3] * (constant[1] - 1) * -1;break;
        case 14:formula5[5] = pow(formula5[3],(constant[1] - 1));break;
        case 15:formula5[5] = pow((constant[1] - 1),formula5[3]);break;
        case 16:formula5[5] = pow(formula5[3],1 / (constant[1] - 1));break;
    }

    switch ( (int)(formula[9] - 1) ) {
        case 0:formula5[10] = formula5[8];break;
        case 1:formula5[10] = fabs(formula5[8]);break;
        case 2:formula5[10] = sin(formula5[8]);break;
        case 3:formula5[10] = cos(formula5[8]);break;
        case 4:formula5[10] = tan(formula5[8]);break;
        case 5:formula5[10] = log(formula5[8]);break;
        case 6:formula5[10] = log10(formula5[8]);break;
        case 7:formula5[10] = formula5[8] + (constant[3] - 1);break;
        case 8:formula5[10] = formula5[8] * (constant[3] - 1);break;
        case 9:formula5[10] = formula5[8] / (constant[3] - 1);break;
        case 10:formula5[10] = (constant[3] - 1) / formula5[8];break;
        case 11:formula5[10] = formula5[8] - (constant[3] - 1);break;
        case 12:formula5[10] = (constant[3] - 1) - formula5[8];break;
        case 13:formula5[10] = formula5[8] * (constant[3] - 1) * -1;break;
        case 14:formula5[10] = pow(formula5[8],(constant[3] - 1));break;
        case 15:formula5[10] = pow((constant[3] - 1),formula5[8]);break;
        case 16:formula5[10] = pow(formula5[8],1 / (constant[3] - 1));break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:formula5[7] = formula5[2] + formula5[5];break;
        case 1:formula5[7] = formula5[2] * formula5[5];break;
        case 2:formula5[7] = formula5[2] - formula5[5];break;
        case 3:formula5[7] = formula5[5] - formula5[2];break;
        case 4:formula5[7] = formula5[2] / formula5[5];break;
        case 5:formula5[7] = formula5[5] / formula5[2];break;
        case 6:formula5[7] = pow(formula5[2],formula5[5]);break;
        case 7:formula5[7] = pow(formula5[5],formula5[2]);break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:formula5[7] = formula5[7];break;
        case 1:formula5[7] = fabs(formula5[7]);break;
        case 2:formula5[7] = sin(formula5[7]);break;
        case 3:formula5[7] = cos(formula5[7]);break;
        case 4:formula5[7] = tan(formula5[7]);break;
        case 5:formula5[7] = log(formula5[7]);break;
        case 6:formula5[7] = log10(formula5[7]);break;
        case 7:formula5[7] = formula5[7] + (constant[2] - 1);break;
        case 8:formula5[7] = formula5[7] * (constant[2] - 1);break;
        case 9:formula5[7] = formula5[7] / (constant[2] - 1);break;
        case 10:formula5[7] = (constant[2] - 1) / formula5[7];break;
        case 11:formula5[7] = formula5[7] - (constant[2] - 1);break;
        case 12:formula5[7] = (constant[2] - 1) - formula5[7];break;
        case 13:formula5[7] = formula5[7] * (constant[2] - 1) * -1;break;
        case 14:formula5[7] = pow(formula5[7],(constant[2] - 1));break;
        case 15:formula5[7] = pow((constant[2] - 1),formula5[7]);break;
        case 16:formula5[7] = pow(formula5[7],1 / (constant[2] - 1));break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:formula5[12] = formula5[7] + formula5[10];break;
        case 1:formula5[12] = formula5[7] * formula5[10];break;
        case 2:formula5[12] = formula5[7] - formula5[10];break;
        case 3:formula5[12] = formula5[10] - formula5[7];break;
        case 4:formula5[12] = formula5[7] / formula5[10];break;
        case 5:formula5[12] = formula5[10] / formula5[7];break;
        case 6:formula5[12] = pow(formula5[7],formula5[10]);break;
        case 7:formula5[12] = pow(formula5[10],formula5[7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:formula5[12] = formula5[12];break;
        case 1:formula5[12] = fabs(formula5[12]);break;
        case 2:formula5[12] = sin(formula5[12]);break;
        case 3:formula5[12] = cos(formula5[12]);break;
        case 4:formula5[12] = tan(formula5[12]);break;
        case 5:formula5[12] = log(formula5[12]);break;
        case 6:formula5[12] = log10(formula5[12]);break;
        case 7:formula5[12] = formula5[12] + (constant[4] - 1);break;
        case 8:formula5[12] = formula5[12] * (constant[4] - 1);break;
        case 9:formula5[12] = formula5[12] / (constant[4] - 1);break;
        case 10:formula5[12] = (constant[4] - 1) / formula5[12];break;
        case 11:formula5[12] = formula5[12] - (constant[4] - 1);break;
        case 12:formula5[12] = (constant[4] - 1) - formula5[12];break;
        case 13:formula5[12] = formula5[12] * (constant[4] - 1) * -1;break;
        case 14:formula5[12] = pow(formula5[12],(constant[4] - 1));break;
        case 15:formula5[12] = pow((constant[4] - 1),formula5[12]);break;
        case 16:formula5[12] = pow(formula5[12],1 / (constant[4] - 1));break;
    }

if(fabs(formula5[12] - formula5[13]) < 0.0001){
    if(IsStand == 1)
    {Check();} //Check for the analog of the 1 formula

    if(IsCheck == 1)
    {
    IsCheck = 0;
    Print();
    Print2();
    Print3();
    PrintForm(); // Found 2 formula, not analog
    printf("\n———\n");
    getch();
    }

    if (IsStand == 0) // Found 1 formula
    {
    Print();
    Print2();
    Print3();
    PrintForm();
    printf("\n~~~~~~\n");
    getch();
    Standart(); // Save the values of 1 formula
    }
}
}

void Print()
{
    int i;
    printf("\n");
    for(i=0;i!=14;i++)
    {
    if(i!= 1 && i!= 4 && i!= 6 && i!= 9 && i!= 11)
    printf("%f ",formula[i]);
    else
    printf("%f ",formula[i] - 1);
    }
    printf("\n");
    for(i=0;i!=5;i++)
    {
    printf("%f ",constant[i] - 1);
    }
    printf("\n");
}

void Print2()
{
    int i;
    printf("\n");
    for(i=0;i!=14;i++)
    {
    if(i!= 1 && i!= 4 && i!= 6 && i!= 9 && i!= 11)
    printf("%f ",formula2[i]);
    else
    printf("%f ",formula2[i] - 1);
    }
    printf("\n");
    for(i=0;i!=5;i++)
    {
    printf("%f ",constant[i] - 1);
    }
    printf("\n");
}

void Print3()
{
    int i;
    printf("\n");
    for(i=0;i!=14;i++)
    {
```

```
    if(i!= 1 && i!= 4 && i!= 6 && i!= 9 && i!= 11)
    printf("%f ",formula3[i]);
    else
    printf("%f ",formula3[i] - 1);
    }
    printf("\n");
    for(i=0;i!=5;i++)
    {
    printf("%f ",constant[i] - 1);
    }
    printf("\n");
}

void PrintForm()
{
    int i, caseo, caseo2;
    printf("\n");

switch ( (int)(formula[1] - 1) ) {
        case 0:printf("x");break;
        case 1:printf("|x|");break;
        case 2:printf("sinx");break;
        case 3:printf("cosx");break;
        case 4:printf("tanx");break;
        case 5:printf("logx");break;
        case 6:printf("log10x");break;
        case 7:printf("x + %f",constant[0] - 1);break;
        case 8:printf("x * %f",constant[0] - 1);break;
        case 9:printf("x / %f",constant[0] - 1);break;
        case 10:printf("x /0 %f",constant[0] - 1);break;
        case 11:printf("x - %f",constant[0] - 1);break;
        case 12:printf("x -0 %f",constant[0] - 1);break;
        case 13:printf("x * (-)%f",constant[0] - 1);;break;
        case 14:printf("x ^ %f",constant[0] - 1);break;
        case 15:printf("x ^0 %f",constant[0] - 1);break;
        case 16:printf("x ^ 1/%f",constant[0] - 1);break;
    }

switch ( (int)(formula[4] - 1) ) {
        case 0:printf(" y");break;
        case 1:printf(" |y|");break;
        case 2:printf(" siny");break;
        case 3:printf(" cosy");break;
        case 4:printf(" tany");break;
        case 5:printf(" logy");break;
        case 6:printf(" log10y");break;
        case 7:printf(" y + %f",constant[1] - 1);break;
        case 8:printf(" y * %f",constant[1] - 1);break;
        case 9:printf(" y / %f",constant[1] - 1);break;
        case 10:printf(" y /0 %f",constant[1] - 1);break;
        case 11:printf(" y - %f",constant[1] - 1);break;
        case 12:printf(" y -0 %f",constant[1] - 1);break;
        case 13:printf(" y * (-)%f",constant[1] - 1);;break;
        case 14:printf(" y ^ %f",constant[1] - 1);break;
        case 15:printf(" y ^0 %f",constant[1] - 1);break;
        case 16:printf(" y ^ 1/%f",constant[1] - 1);break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;
    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:printf(" (+)");break;
        case 1:printf(" (*)");break;
        case 2:printf(" (-)");break;
        case 3:printf(" (-0)");break;
        case 4:printf(" (/)");break;
        case 5:printf(" (/0)");break;
        case 6:printf(" (^)");break;
        case 7:printf(" (^0)");break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:printf("");break;
        case 1:printf("||");break;
        case 2:printf("sin");break;
        case 3:printf("cos");break;
        case 4:printf("tan");break;
        case 5:printf("log");break;
        case 6:printf("log10");break;
        case 7:printf(" + %f",constant[2] - 1);break;
        case 8:printf(" * %f",constant[2] - 1);break;
        case 9:printf(" / %f",constant[2] - 1);break;
        case 10:printf(" /0 %f",constant[2] - 1);break;
        case 11:printf(" - %f",constant[2] - 1);break;
        case 12:printf(" -0 %f",constant[2] - 1);break;
        case 13:printf(" * (-)%f",constant[2] - 1);break;
        case 14:printf(" ^ %f",constant[2] - 1);break;
        case 15:printf(" ^0 %f",constant[2] - 1);break;
        case 16:printf(" ^ 1/%f",constant[2] - 1);break;
    }

    switch ( (int)(formula[9] - 1) ) {
        case 0:printf(" z");break;
        case 1:printf(" |z|");break;
        case 2:printf(" sinz");break;
        case 3:printf(" cosz");break;
        case 4:printf(" tanz");break;
        case 5:printf(" logz");break;
        case 6:printf(" log10z");break;
        case 7:printf(" z + %f",constant[3] - 1);break;
        case 8:printf(" z * %f",constant[3] - 1);break;
        case 9:printf(" z / %f",constant[3] - 1);break;
        case 10:printf(" z /0 %f",constant[3] - 1);break;
        case 11:printf(" z - %f",constant[3] - 1);break;
        case 12:printf(" z -0 %f",constant[3] - 1);break;
        case 13:printf(" z * (-)%f",constant[3] - 1);;break;
        case 14:printf(" z ^ %f",constant[3] - 1);break;
        case 15:printf(" z ^0 %f",constant[3] - 1);break;
        case 16:printf(" z ^ 1/%f",constant[3] - 1);break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:printf(" (+)");break;
        case 1:printf(" (*)");break;
        case 2:printf(" (-)");break;
        case 3:printf(" (-0)");break;
        case 4:printf(" (/)");break;
        case 5:printf(" (/0)");break;
        case 6:printf(" (^)");break;
        case 7:printf(" (^0)");break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:printf("");break;
        case 1:printf("||");break;
        case 2:printf("sin");break;
        case 3:printf("cos");break;
        case 4:printf("tan");break;
        case 5:printf("log");break;
        case 6:printf("log10");break;
        case 7:printf(" + %f",constant[4] - 1);break;
        case 8:printf(" * %f",constant[4] - 1);break;
        case 9:printf(" / %f",constant[4] - 1);break;
        case 10:printf(" /0 %f",constant[4] - 1);break;
        case 11:printf(" - %f",constant[4] - 1);break;
        case 12:printf(" -0 %f",constant[4] - 1);break;
        case 13:printf(" * (-)%f",constant[4] - 1);break;
        case 14:printf(" ^ %f",constant[4] - 1);break;
        case 15:printf(" ^0 %f",constant[4] - 1);break;
        case 16:printf(" ^ 1/%f",constant[4] - 1);break;
    }
}

void Check() //Check for the analog of the 1 formula
{
    int c,i,j, caseo, caseo2;
    for(c=0;c!=20;c++)
    {
    check[c][0] = formula[0] + (double)c;
    check[c][3] = formula[3] + (double)c;
    check[c][8] = formula[8] + (double)c;
    }

    for(i=0;i!=3;i++)//!=8
    {

    switch ( (int)(formula[1] - 1) ) {
        case 0:check[i][2] = check[i][0];break;
        case 1:check[i][2] = fabs(check[i][0]);break;
        case 2:check[i][2] = sin(check[i][0]);break;
        case 3:check[i][2] = cos(check[i][0]);break;
        case 4:check[i][2] = tan(check[i][0]);break;
        case 5:check[i][2] = log(check[i][0]);break;
        case 6:check[i][2] = log10(check[i][0]);break;
        case 7:check[i][2] = check[i][0] + (constant[0] - 1);break;
        case 8:check[i][2] = check[i][0] * (constant[0] - 1);break;
        case 9:check[i][2] = check[i][0] / (constant[0] - 1);break;
        case 10:check[i][2] = (constant[0] - 1) / check[i][0];break;
        case 11:check[i][2] = check[i][0] - (constant[0] - 1);break;
        case 12:check[i][2] = (constant[0] - 1) - check[i][0];break;
        case 13:check[i][2] = check[i][0] * (constant[0] - 1) * -1;break;
        case 14:check[i][2] = pow(check[i][0],(constant[0] - 1));break;
        case 15:check[i][2] = pow((constant[0] - 1),check[i][0]);break;
        case 16:check[i][2] = pow(check[i][0],1 / (constant[0] - 1));break;
    }

switch ( (int)(formula[4] - 1) ) {
        case 0:check[i][5] = check[i][3];break;
        case 1:check[i][5] = fabs(check[i][3]);break;
        case 2:check[i][5] = sin(check[i][3]);break;
        case 3:check[i][5] = cos(check[i][3]);break;
        case 4:check[i][5] = tan(check[i][3]);break;
        case 5:check[i][5] = log(check[i][3]);break;
        case 6:check[i][5] = log10(check[i][3]);break;
        case 7:check[i][5] = check[i][3] + (constant[1] - 1);break;
        case 8:check[i][5] = check[i][3] * (constant[1] - 1);break;
        case 9:check[i][5] = check[i][3] / (constant[1] - 1);break;
        case 10:check[i][5] = (constant[1] - 1) / check[i][3];break;
        case 11:check[i][5] = check[i][3] - (constant[1] - 1);break;
        case 12:check[i][5] = (constant[1] - 1) - check[i][3];break;
        case 13:check[i][5] = check[i][3] * (constant[1] - 1) * -1;break;
        case 14:check[i][5] = pow(check[i][3],(constant[1] - 1));break;
        case 15:check[i][5] = pow((constant[1] - 1),check[i][3]);break;
        case 16:check[i][5] = pow(check[i][3],1 / (constant[1] - 1));break;
    }

    switch ( (int)(formula[9] - 1) ) {
        case 0:check[i][10] = check[i][8];break;
        case 1:check[i][10] = fabs(check[i][8]);break;
        case 2:check[i][10] = sin(check[i][8]);break;
        case 3:check[i][10] = cos(check[i][8]);break;
        case 4:check[i][10] = tan(check[i][8]);break;
        case 5:check[i][10] = log(check[i][8]);break;
        case 6:check[i][10] = log10(check[i][8]);break;
        case 7:check[i][10] = check[i][8] + (constant[3] - 1);break;
        case 8:check[i][10] = check[i][8] * (constant[3] - 1);break;
        case 9:check[i][10] = check[i][8] / (constant[3] - 1);break;
        case 10:check[i][10] = (constant[3] - 1) / check[i][8];break;
        case 11:check[i][10] = check[i][8] - (constant[3] - 1);break;
        case 12:check[i][10] = (constant[3] - 1) - check[i][8];break;
        case 13:check[i][10] = check[i][8] * (constant[3] - 1) * -1;break;
        case 14:check[i][10] = pow(check[i][8],(constant[3] - 1));break;
        case 15:check[i][10] = pow((constant[3] - 1),check[i][8]);break;
        case 16:check[i][10] = pow(check[i][8],1 / (constant[3] - 1));break;
    }

    caseo = formula[6] - 1;
    caseo2 = formula[11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:check[i][7] = check[i][2] + check[i][5];break;
        case 1:check[i][7] = check[i][2] * check[i][5];break;
        case 2:check[i][7] = check[i][2] - check[i][5];break;
        case 3:check[i][7] = check[i][5] - check[i][2];break;
        case 4:check[i][7] = check[i][2] / check[i][5];break;
        case 5:check[i][7] = check[i][5] / check[i][2];break;
        case 6:check[i][7] = pow(check[i][2],check[i][5]);break;
        case 7:check[i][7] = pow(check[i][5],check[i][2]);break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:check[i][7] = check[i][7];break;
        case 1:check[i][7] = fabs(check[i][7]);break;
        case 2:check[i][7] = sin(check[i][7]);break;
        case 3:check[i][7] = cos(check[i][7]);break;
        case 4:check[i][7] = tan(check[i][7]);break;
        case 5:check[i][7] = log(check[i][7]);break;
        case 6:check[i][7] = log10(check[i][7]);break;
        case 7:check[i][7] = check[i][7] + (constant[2] - 1);break;
        case 8:check[i][7] = check[i][7] * (constant[2] - 1);break;
        case 9:check[i][7] = check[i][7] / (constant[2] - 1);break;
        case 10:check[i][7] = (constant[2] - 1) / check[i][7];break;
        case 11:check[i][7] = check[i][7] - (constant[2] - 1);break;
        case 12:check[i][7] = (constant[2] - 1) - check[i][7];break;
        case 13:check[i][7] = check[i][7] * (constant[2] - 1) * -1;break;
        case 14:check[i][7] = pow(check[i][7],(constant[2] - 1));break;
        case 15:check[i][7] = pow((constant[2] - 1),check[i][7]);break;
        case 16:check[i][7] = pow(check[i][7],1 / (constant[2] - 1));break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:check[i][12] = check[i][7] + check[i][10];break;
        case 1:check[i][12] = check[i][7] * check[i][10];break;
        case 2:check[i][12] = check[i][7] - check[i][10];break;
        case 3:check[i][12] = check[i][10] - check[i][7];break;
        case 4:check[i][12] = check[i][7] / check[i][10];break;
        case 5:check[i][12] = check[i][10] / check[i][7];break;
        case 6:check[i][12] = pow(check[i][7],check[i][10]);break;
        case 7:check[i][12] = pow(check[i][10],check[i][7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:check[i][12] = check[i][12];break;
        case 1:check[i][12] = fabs(check[i][12]);break;
        case 2:check[i][12] = sin(check[i][12]);break;
        case 3:check[i][12] = cos(check[i][12]);break;
        case 4:check[i][12] = tan(check[i][12]);break;
        case 5:check[i][12] = log(check[i][12]);break;
        case 6:check[i][12] = log10(check[i][12]);break;
        case 7:check[i][12] = check[i][12] + (constant[4] - 1);break;
        case 8:check[i][12] = check[i][12] * (constant[4] - 1);break;
        case 9:check[i][12] = check[i][12] / (constant[4] - 1);break;
        case 10:check[i][12] = (constant[4] - 1) / check[i][12];break;
        case 11:check[i][12] = check[i][12] - (constant[4] - 1);break;
        case 12:check[i][12] = (constant[4] - 1) - check[i][12];break;
        case 13:check[i][12] = check[i][12] * (constant[4] - 1) * -1;break;
        case 14:check[i][12] = pow(check[i][12],(constant[4] - 1));break;
        case 15:check[i][12] = pow((constant[4] - 1),check[i][12]);break;
        case 16:check[i][12] = pow(check[i][12],1 / (constant[4] - 1));break;
    }

    switch ( (int)(standart[0][1] - 1) ) {
        case 0:standart[i][2] = standart[i][0];break;
        case 1:standart[i][2] = fabs(standart[i][0]);break;
        case 2:standart[i][2] = sin(standart[i][0]);break;
        case 3:standart[i][2] = cos(standart[i][0]);break;
        case 4:standart[i][2] = tan(standart[i][0]);break;
        case 5:standart[i][2] = log(standart[i][0]);break;
        case 6:standart[i][2] = log10(standart[i][0]);break;
        case 7:standart[i][2] = standart[i][0] + (stconstant[0] - 1);break;
        case 8:standart[i][2] = standart[i][0] * (stconstant[0] - 1);break;
        case 9:standart[i][2] = standart[i][0] / (stconstant[0] - 1);break;
        case 10:standart[i][2] = (stconstant[0] - 1) / standart[i][0];break;
        case 11:standart[i][2] = standart[i][0] - (stconstant[0] - 1);break;
        case 12:standart[i][2] = (stconstant[0] - 1) - standart[i][0];break;
        case 13:standart[i][2] = standart[i][0] * (stconstant[0] - 1) * -1;break;
        case 14:standart[i][2] = pow(standart[i][0],(stconstant[0] - 1));break;
        case 15:standart[i][2] = pow((stconstant[0] - 1),standart[i][0]);break;
        case 16:standart[i][2] = pow(standart[i][0],1 / (stconstant[0] - 1));break;
    }

switch ( (int)(standart[0][4] - 1) ) {
        case 0:standart[i][5] = standart[i][3];break;
        case 1:standart[i][5] = fabs(standart[i][3]);break;
        case 2:standart[i][5] = sin(standart[i][3]);break;
        case 3:standart[i][5] = cos(standart[i][3]);break;
        case 4:standart[i][5] = tan(standart[i][3]);break;
        case 5:standart[i][5] = log(standart[i][3]);break;
        case 6:standart[i][5] = log10(standart[i][3]);break;
        case 7:standart[i][5] = standart[i][3] + (stconstant[1] - 1);break;
        case 8:standart[i][5] = standart[i][3] * (stconstant[1] - 1);break;
        case 9:standart[i][5] = standart[i][3] / (stconstant[1] - 1);break;
        case 10:standart[i][5] = (stconstant[1] - 1) / standart[i][3];break;
        case 11:standart[i][5] = standart[i][3] - (stconstant[1] - 1);break;
        case 12:standart[i][5] = (stconstant[1] - 1) - standart[i][3];break;
        case 13:standart[i][5] = standart[i][3] * (stconstant[1] - 1) * -1;break;
        case 14:standart[i][5] = pow(standart[i][3],(stconstant[1] - 1));break;
        case 15:standart[i][5] = pow((stconstant[1] - 1),standart[i][3]);break;
        case 16:standart[i][5] = pow(standart[i][3],1 / (stconstant[1] - 1));break;
    }

    switch ( (int)(standart[0][9] - 1) ) {
        case 0:standart[i][10] = standart[i][8];break;
        case 1:standart[i][10] = fabs(standart[i][8]);break;
        case 2:standart[i][10] = sin(standart[i][8]);break;
        case 3:standart[i][10] = cos(standart[i][8]);break;
        case 4:standart[i][10] = tan(standart[i][8]);break;
        case 5:standart[i][10] = log(standart[i][8]);break;
        case 6:standart[i][10] = log10(standart[i][8]);break;
        case 7:standart[i][10] = standart[i][8] + (stconstant[3] - 1);break;
        case 8:standart[i][10] = standart[i][8] * (stconstant[3] - 1);break;
        case 9:standart[i][10] = standart[i][8] / (stconstant[3] - 1);break;
        case 10:standart[i][10] = (stconstant[3] - 1) / standart[i][8];break;
        case 11:standart[i][10] = standart[i][8] - (stconstant[3] - 1);break;
        case 12:standart[i][10] = (stconstant[3] - 1) - standart[i][8];break;
        case 13:standart[i][10] = standart[i][8] * (stconstant[3] - 1) * -1;break;
        case 14:standart[i][10] = pow(standart[i][8],(stconstant[3] - 1));break;
        case 15:standart[i][10] = pow((stconstant[3] - 1),standart[i][8]);break;
        case 16:standart[i][10] = pow(standart[i][8],1 / (stconstant[3] - 1));break;
    }

    caseo = standart[0][6] - 1;
    caseo2 = standart[0][11] - 1;

    if(caseo > 7){caseo = caseo - 8;constf = caseo - floor(caseo/17)*17; caseo = floor(caseo/17);
else flagc1 = 0;
switch ( caseo ) {
        case 0:standart[i][7] = standart[i][2] + standart[i][5];break;
        case 1:standart[i][7] = standart[i][2] * standart[i][5];break;
        case 2:standart[i][7] = standart[i][2] - standart[i][5];break;
        case 3:standart[i][7] = standart[i][5] - standart[i][2];break;
        case 4:standart[i][7] = standart[i][2] / standart[i][5];break;
        case 5:standart[i][7] = standart[i][5] / standart[i][2];break;
        case 6:standart[i][7] = pow(standart[i][2],standart[i][5]);break;
        case 7:standart[i][7] = pow(standart[i][5],standart[i][2]);break;
    }

if(flagc1 == 1)
switch ( constf ) {
        case 0:standart[i][7] = standart[i][7];break;
        case 1:standart[i][7] = fabs(standart[i][7]);break;
        case 2:standart[i][7] = sin(standart[i][7]);break;
        case 3:standart[i][7] = cos(standart[i][7]);break;
        case 4:standart[i][7] = tan(standart[i][7]);break;
        case 5:standart[i][7] = log(standart[i][7]);break;
        case 6:standart[i][7] = log10(standart[i][7]);break;
        case 7:standart[i][7] = standart[i][7] + (stconstant[2] - 1);break;
        case 8:standart[i][7] = standart[i][7] * (stconstant[2] - 1);break;
        case 9:standart[i][7] = standart[i][7] / (stconstant[2] - 1);break;
        case 10:standart[i][7] = (stconstant[2] - 1) / standart[i][7];break;
        case 11:standart[i][7] = standart[i][7] - (stconstant[2] - 1);break;
        case 12:standart[i][7] = (stconstant[2] - 1) - standart[i][7];break;
        case 13:standart[i][7] = standart[i][7] * (stconstant[2] - 1) * -1;break;
        case 14:standart[i][7] = pow(standart[i][7],(stconstant[2] - 1));break;
        case 15:standart[i][7] = pow((stconstant[2] - 1),standart[i][7]);break;
        case 16:standart[i][7] = pow(standart[i][7],1 / (stconstant[2] - 1));break;
    }

if(caseo2 > 7){caseo2 = caseo2 - 8;constf = caseo2 - floor(caseo2/17)*17; caseo2 = floor(caseo2/1
else flagc2 = 0;
switch ( caseo2 ) {
        case 0:standart[i][12] = standart[i][7] + standart[i][10];break;
        case 1:standart[i][12] = standart[i][7] * standart[i][10];break;
        case 2:standart[i][12] = standart[i][7] - standart[i][10];break;
        case 3:standart[i][12] = standart[i][10] - standart[i][7];break;
        case 4:standart[i][12] = standart[i][7] / standart[i][10];break;
        case 5:standart[i][12] = standart[i][10] / standart[i][7];break;
        case 6:standart[i][12] = pow(standart[i][7],standart[i][10]);break;
        case 7:standart[i][12] = pow(standart[i][10],standart[i][7]);break;
    }
if(flagc2 == 1)
switch ( constf ) {
        case 0:standart[i][12] = standart[i][12];break;
        case 1:standart[i][12] = fabs(standart[i][12]);break;
        case 2:standart[i][12] = sin(standart[i][12]);break;
        case 3:standart[i][12] = cos(standart[i][12]);break;
        case 4:standart[i][12] = tan(standart[i][12]);break;
        case 5:standart[i][12] = log(standart[i][12]);break;
        case 6:standart[i][12] = log10(standart[i][12]);break;
        case 7:standart[i][12] = standart[i][12] + (stconstant[4] - 1);break;
        case 8:standart[i][12] = standart[i][12] * (stconstant[4] - 1);break;
        case 9:standart[i][12] = standart[i][12] / (stconstant[4] - 1);break;
        case 10:standart[i][12] = (stconstant[4] - 1) / standart[i][12];break;
        case 11:standart[i][12] = standart[i][12] - (stconstant[4] - 1);break;
        case 12:standart[i][12] = (stconstant[4] - 1) - standart[i][12];break;
        case 13:standart[i][12] = standart[i][12] * (stconstant[4] - 1) * -1;break;
        case 14:standart[i][12] = pow(standart[i][12],(stconstant[4] - 1));break;
        case 15:standart[i][12] = pow((stconstant[4] - 1),standart[i][12]);break;
        case 16:standart[i][12] = pow(standart[i][12],1 / (stconstant[4] - 1));break;
    }

    if( fabs(standart[i][12] - check[i][12]) > 0.0001 ){
    printf("=====");

    printf("\n");
    for(j=0;j!=13;j++)
    {
    if(j!= 1 && j!= 4 && j!= 6 && j!= 9 && j!= 11)
    printf("%f ",check[i][j]);
    else
    printf("%f ",check[i][j] - 1);
    }
    printf("\n");
    for(j=0;j!=5;j++)
    {
    printf("%f ",constant[j] - 1);
    }
    printf("\n");

    printf("=====");
    printf("\n");
    for(j=0;j!=13;j++)
    {
    if(j!= 1 && j!= 4 && j!= 6 && j!= 9 && j!= 11)
    printf("%f ",standart[i][j]);
    else
    printf("%f ",standart[i][j] - 1);
    }
    printf("\n");
    for(j=0;j!=5;j++)
    {
    printf("%f ",stconstant[j] - 1);
    }
    printf("\n");
    IsCheck = 1;
    }
    }
}

void Standart() // Save the values of 1 formula
{
    int s;
    for(s=0;s!=20;s++)
    {
    standart[s][0] = formula[0] + (double)s;
    standart[s][1] = formula[1];
    standart[s][2] = formula[2];
    standart[s][3] = formula[3] + (double)s;
    standart[s][4] = formula[4];
    standart[s][5] = formula[5];
    standart[s][6] = formula[6];
    standart[s][7] = formula[7];
    standart[s][8] = formula[8] + (double)s;
    standart[s][9] = formula[9];
    standart[s][10] = formula[10];
    standart[s][11] = formula[11];
    standart[s][12] = formula[12];
    }
    stconstant[0] = constant[0];
    stconstant[1] = constant[1];
    stconstant[2] = constant[2];
    stconstant[3] = constant[3];
    stconstant[4] = constant[4];

    IsStand = 1;
    Print3();
    printf("\n standart \n");
```

```
}

void Scan()
{
    scanf("%lf",&formula[0]);
    scanf("%lf",&formula[3]);
    scanf("%lf",&formula[8]);
    scanf("%lf",&formula[13]);
    scanf("%lf",&formula2[0]);
    scanf("%lf",&formula2[3]);
    scanf("%lf",&formula2[8]);
    scanf("%lf",&formula2[13]);
    scanf("%lf",&formula3[0]);
    scanf("%lf",&formula3[3]);
    scanf("%lf",&formula3[8]);
    scanf("%lf",&formula3[13]);
    scanf("%lf",&formula4[0]);
    scanf("%lf",&formula4[3]);
    scanf("%lf",&formula4[8]);
    scanf("%lf",&formula4[13]);
    scanf("%lf",&formula5[0]);
    scanf("%lf",&formula5[3]);
    scanf("%lf",&formula5[8]);
    scanf("%lf",&formula5[13]);
}
```

Listing 2: Sequence with many parameters

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#define MAX 5

short func[200][30],mas[200][MAX-2];
short Pcount = 0;
short i = 1,j,index=0,tmp=0;

void main()
{
    do
    {
        for(j=i;j!=MAX;j++)
        {
            for(tmp=0;tmp!=MAX+10;tmp++)
            {
                func[index][0] = i;
                func[index][1] = j+1;
                func[index][2] = 9;
                index++;
            }
        }
        i++;
    }
    while(i!=MAX);

    if(MAX-2!=1)
    {
```

```
for(index=0;index!=10;index++)
{
    j=1;
    for(i=1;i!=MAX+1;i++)
        if(func[index*15][0]!=i && func[index*15][1]!=i)
                {
                    mas[index][j]=i;
                    j++;
                }
}
for(index=0;index!=10;index++)
{
    for(i=1;i!=MAX-2;i++)
    {
        for(j=i;j!=MAX-2;j++)
        {
                func[index*15+j*3+i*3-6][3] = mas[index][i];
                func[index*15+j*3+i*3-6][4] = mas[index][j+1];
                func[index*15+j*3+i*3-6][5] = 9;
                if(2!=0)
                    {
                        for(tmp=0;tmp!=MAX-1;tmp++)
                            {
                                if(tmp!=i && tmp!=j+1)
                                    {
                                    func[index*15+j*3+i*3-6][6] = mas[index][tmp];
                                    func[index*15+j*3+i*3-6][7] = 9;
                                    func[index*15+j*3+i*3-6][8] = 9;

                                    func[index*15+j*3+i*3-5][3] = mas[index][i];
                                    func[index*15+j*3+i*3-5][4] = mas[index][j+1];
                                    func[index*15+j*3+i*3-5][5] = 9;
                                    func[index*15+j*3+i*3-5][6] = 9;
                                    func[index*15+j*3+i*3-5][7] = mas[index][tmp];
                                    func[index*15+j*3+i*3-5][8] = 9;

                                    func[index*15+j*3+i*3-4][3] = mas[index][tmp];
                                    func[index*15+j*3+i*3-4][4] = 9;
                                    func[index*15+j*3+i*3-4][5] = mas[index][i];
                                    func[index*15+j*3+i*3-4][6] = mas[index][j+1];
                                    func[index*15+j*3+i*3-4][7] = 9;
                                    func[index*15+j*3+i*3-4][8] = 9;
                                    }
                            }
                    }
        }
    }
}
        for(index=0;index!=10;index++)
        for(i=1;i!=MAX-1;i++)
        {
            func[index*15+8+i][3] = mas[index][i];func[index*15+8+i][4] =9;
            if(i==1){func[index*15+8+i][5] = mas[index][i+1];func[index*15+8+i][6] =9;
            func[index*15+8+i][7] = mas[index][i+2];func[index*15+8+i][8] =9;}
            if(i==2){func[index*15+8+i][5] = mas[index][i-1];func[index*15+8+i][6] =9;
            func[index*15+8+i][7] = mas[index][i+1];func[index*15+8+i][8] =9;}
            if(i==3){func[index*15+8+i][5] = mas[index][i-1];func[index*15+8+i][6] =9;
            func[index*15+8+i][7] = mas[index][i-2];func[index*15+8+i][8] =9;}

            func[index*15+11+i][3] = mas[index][i];func[index*15+11+i][4] =9;
            if(i==1){func[index*15+11+i][5] = mas[index][i+2];func[index*15+11+i][6] =9;
```

```
                    func[index*15+11+i][7] = mas[index][i+1];func[index*15+11+i][8] =9;}
                    if(i==2){func[index*15+11+i][5] = mas[index][i+1];func[index*15+11+i][6] =9;
                    func[index*15+11+i][7] = mas[index][i-1];func[index*15+11+i][8] =9;}
                    if(i==3){func[index*15+11+i][5] = mas[index][i-2];func[index*15+11+i][6] =9;
                    func[index*15+11+i][7] = mas[index][i-1];func[index*15+11+i][8] =9;}
                }
    }
Print();
}
void Print()
{
short n1 = 0, n2 = 0;

    for(i=0;i!=151;i++)
{
    n1 = func[i][0];
    n2 = func[i][1];
    for(j=i;j!=151;j++)
    {
    if((func[j][3] == n1 && func[j][4] == n2)||(func[j][4] == n1 && func[j][5] == n2)||(func[j][5]
        {
            func[j][0] = 0;
        }
    }
}
    for(i=0;i!=151;i++)
{
    if(func[i][0] != 0){
    for(j=0;func[i][j]!=0;j++)
    {
    printf("%d",func[i][j]);
    }
    printf("\n");}
}
     for(i=0;i<10;i++)
{
    for(j=1;j<MAX-1;j++)
    {
    printf("%d",mas[i][j]);
    }
    printf("\n");
}
printf("\n 9 is a operation \n");
getch();
}
}
```