public class LinearEquation {

public String a, former, latter;

public char x;

void input(String eq) {

a = eq;

}

void check() {

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

if (a.charAt(i) != '=' && a.charAt(i) != '+' && a.charAt(i) != '-' && a.charAt(i) != '/'

&& a.charAt(i) != '\*' && a.charAt(i) != '(' && a.charAt(i) != ')' && a.charAt(i) != '{'

&& a.charAt(i) != '}' && a.charAt(i) != '[' && a.charAt(i) != ']'

&& !Character.isWhitespace(a.charAt(i)) && !Character.isLetterOrDigit(a.charAt(i)))

endIt(1);

boolean flag = true;

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

if (a.charAt(i) == '=')

flag = false;

if (flag)

endIt(2);

flag = true;

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

if (Character.isLetter(a.charAt(i))) {

flag = false;

x = a.charAt(i);

}

if (flag)

endIt(3);

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

if (Character.isLetter(a.charAt(i)) && a.charAt(i) != x)

endIt(4);

}

void initializer() {

String mod = "";

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

if (!Character.isWhitespace(a.charAt(i)))

mod += a.charAt(i);

a=mod;

mod = "";

former = a.substring(0, a.indexOf('='));

latter = a.substring(a.indexOf('='));

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

if (latter.charAt(i) != '=')

mod += latter.charAt(i);

latter = mod;

mod = "";

if (former.equals(""))

former = "0";

if (latter.equals(""))

latter = "0";

a = former + '=' + latter;

}

String bracketHolder(String str) {

int j = 0;

String mod = "";

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

if (str.charAt(i) == '(' || str.charAt(i) == '{' || str.charAt(i) == '[') {

mod += '(';

j++;

} else if (str.charAt(i) == ')' || str.charAt(i) == '}' || str.charAt(i) == ']') {

mod += ')';

j--;

} else

mod += str.charAt(i);

if (j < 0)

endIt(5);

}

if (j != 0)

endIt(6);

str = mod;

return str;

}

String symbolHolderWrap(char a, char b) {

String ret = "";

ret += b;

if (Character.isDigit(b)) {

if (Character.isLetter(a) || a == ')')

ret = "\*" + ret;

}

if (Character.isLetter(b) || b == '(') {

if (Character.isLetterOrDigit(a) || a == ')')

ret = "\*" + ret;

if (a == '.')

ret = "0\*" + ret;

}

if (b == ')') {

if (a == '.' || a == '-' || a == '+')

ret = "0" + ret;

if (a == '\*' || a == '/' || a == '(')

ret = "1" + ret;

}

if (b == '.') {

ret = "";

if (a == ')' || Character.isLetter(a))

ret = "\*";

else if (!Character.isDigit(a) && a != '.')

ret += "0.";

else

ret = ".";

}

if (b == '+' || b == '-') {

if (a == '.' || a == '-' || a == '+')

ret = "0" + ret;

if (a == '\*' || a == '/')

ret = "1" + ret;

if (a == '(')

ret = "(0" + ret + "1)\*";

}

if (b == '/' || b == '\*') {

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

ret = "1" + ret;

if (a == '.')

ret = "0" + ret;

}

return ret;

}

String symbolHolder(String str) {

char[] carr = new char[str.length() + 2];

String[] sarr = new String[carr.length];

carr[0] = '(';

carr[carr.length - 1] = ')';

sarr[0] = ")";

for (int i = 1; i < carr.length - 1; i++)

carr[i] = str.charAt(i - 1);

for (int i = 1; i < carr.length; i++) {

sarr[i] = symbolHolderWrap(carr[i - 1], carr[i]);

if (sarr[i].equals("remove"))

carr[i] = carr[i - 1];

else

carr[i] = sarr[i].charAt(sarr[i].length() - 1);

}

str = "";

for (int i = 1; i < sarr.length; i++)

if (!sarr[i].equals("remove"))

str += sarr[i];

return str.substring(0, str.length() - 1);

}

String[][] segmentSorter(String str) {

char[] carr = new char[str.length()];

String[] value = new String[carr.length];

String[] location = new String[carr.length];

String[] locationend = new String[carr.length];

String[][] ret = new String[4][carr.length + 1];

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

carr[i] = str.charAt(i);

value[i] = "()";

location[i] = "";

locationend[i] = "";

ret[3][i] = "";

}

int j = 0;

int k;

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

if (carr[i] == '(') {

j++;

location[j - 1] += String.valueOf(i);

}

if (carr[i] == ')') {

k = j;

while (true) {

k--;

if (locationend[k].equals("")) {

locationend[k] += String.valueOf(i);

break;

}

}

}

}

for (int i = 0; i < j; i++)

value[i] = str.substring(Integer.parseInt(location[i]), Integer.parseInt(locationend[i])) + ")";

ret[0] = value;

ret[1] = location;

ret[2] = locationend;

ret[3][0] = String.valueOf(j);

return ret;

}

String[][] termSorter(String str) {

char[] carr = new char[str.length()];

String[] value = new String[carr.length];

String[] location = new String[carr.length];

String[] locationend = new String[carr.length];

String[][] ret = new String[4][carr.length + 1];

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

carr[i] = str.charAt(i);

value[i] = "";

location[i] = "";

locationend[i] = "";

ret[3][i] = "";

}

int j = 0;

boolean flag = false;

int brace = 0;

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

if (carr[i] == '(' || carr[i] == '{') {

brace++;

flag = true;

}

if (carr[i] == ')' || carr[i] == '}') {

brace--;

if (brace == 0)

flag = false;

}

if (!flag || carr[i] == '(' || carr[i] == '{') {

if (i == 0)

location[j] += String.valueOf(i);

if (i != carr.length - 1) {

if (carr[i] == '+' || carr[i] == '-' || carr[i] == '=')

location[j] += String.valueOf(i + 1);

if (carr[i + 1] == '+' || carr[i + 1] == '-' || carr[i + 1] == '=')

locationend[j++] += String.valueOf(i);

} else

locationend[j++] += String.valueOf(i);

}

}

for (int i = 0; i < j - 1; i++)

value[i] = str.substring(Integer.parseInt(location[i]), Integer.parseInt(locationend[i]) + 1);

if (j > 0)

value[j - 1] = str.substring(Integer.parseInt(location[j - 1]));

ret[0] = value;

ret[1] = location;

ret[2] = locationend;

ret[3][0] = String.valueOf(j);

return ret;

}

String[][] constantSorter(String str) {

char[] carr = new char[str.length()];

String[] value = new String[carr.length];

String[] location = new String[carr.length];

String[] locationend = new String[carr.length];

String[][] ret = new String[4][carr.length + 1];

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

carr[i] = str.charAt(i);

value[i] = "";

location[i] = "";

locationend[i] = "";

ret[3][i] = "";

}

int j = 0;

boolean flag = false;

int brace = 0;

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

if (carr[i] == '(' || carr[i] == '{') {

brace++;

flag = true;

}

if (carr[i] == ')' || carr[i] == '}') {

brace--;

if (brace == 0)

flag = false;

}

if (!flag || carr[i] == '(' || carr[i] == '{') {

if (i == 0)

location[j] += String.valueOf(i);

if (i != carr.length - 1) {

if (carr[i] == '/' || carr[i] == '\*')

location[j] += String.valueOf(i + 1);

if (carr[i + 1] == '/' || carr[i + 1] == '\*')

locationend[j++] += String.valueOf(i);

} else

locationend[j++] += String.valueOf(i);

}

}

for (int i = 0; i < j - 1; i++)

value[i] = str.substring(Integer.parseInt(location[i]), Integer.parseInt(locationend[i]) + 1);

if (j > 0)

value[j - 1] = str.substring(Integer.parseInt(location[j - 1]));

ret[0] = value;

ret[1] = location;

ret[2] = locationend;

ret[3][0] = String.valueOf(j);

return ret;

}

String[][] numeralSorter(String str) {

char[] carr = new char[str.length()];

String[] value = new String[carr.length];

String[] location = new String[carr.length];

String[] locationend = new String[carr.length];

String[][] ret = new String[4][carr.length + 1];

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

carr[i] = str.charAt(i);

value[i] = "";

location[i] = "";

locationend[i] = "";

ret[3][i] = "";

}

int j = 0;

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

if (carr[i] == '.' || Character.isDigit(carr[i])) {

value[j] += carr[i];

if (i == 0)

location[j] += String.valueOf(i);

else if (carr[i - 1] != '.' && !Character.isDigit(carr[i - 1]))

location[j] += String.valueOf(i);

if (i == carr.length - 1)

locationend[j++] += String.valueOf(i);

else if (carr[i + 1] != '.' && !Character.isDigit(carr[i + 1]))

locationend[j++] += String.valueOf(i);

}

ret[0] = value;

ret[1] = location;

ret[2] = locationend;

ret[3][0] = String.valueOf(j);

return ret;

}

String decimalHandler(String str) {

String mod = "";

String retnum[][];

int check, nums, num1, num2, numn = 0;

retnum = numeralSorter(str);

nums = Integer.parseInt(retnum[3][0]);

for (int i = 0; i < nums; i++) {

mod = "";

check = 0;

for (int j = 0; j < retnum[0][i].length(); j++)

if (retnum[0][i].charAt(j) == '.') {

check++;

if (check <= 1)

mod += retnum[0][i].charAt(j);

} else

mod += retnum[0][i].charAt(j);

retnum[0][i] = mod;

}

for (int i = 0; i < nums; i++) {

num1 = Integer.parseInt(retnum[1][i]);

num2 = Integer.parseInt(retnum[2][i]);

if (i == 0)

mod = str.substring(0, num1) + retnum[0][i];

else

mod += str.substring(numn + 1, num1) + retnum[0][i];

numn = num2;

}

if (numn < str.length() - 1)

mod += str.substring(numn + 1);

str = mod;

return str;

}

String termManager(String str) {

String mod = "";

boolean flag = false;

str = "(" + str + ")";

String[][] retseg, retterm, retcon;

int seg1, seg2, segs, term1, term2, termn = 0, terms, con, cons, k, capture = 0;

String path1, path2, mod2, mod3 = "", ret = "";

boolean[] check, check2, check3;

segs = Integer.parseInt(segmentSorter(str)[3][0]);

check2 = new boolean[segs];

check3 = new boolean[segs];

for (int i = 0; i < segs; i++) {

check2[i] = false;

check3[i] = false;

}

while (true) {

flag = true;

for (int i = 0; i < segs; i++)

if (!check2[i])

flag = false;

if (flag)

break;

for (int i = 0; i < segs; i++) {

check2[i] = true;

mod = segmentSorter(str)[0][i];

mod = mod.substring(1, mod.length() - 1);

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

if (mod.charAt(j) == '(') {

for (k = 0; k < segs; k++)

if ((j + 1 + Integer.parseInt(segmentSorter(str)[1][i])) == Integer

.parseInt(segmentSorter(str)[1][k]))

capture = k;

if (!check2[capture])

check2[i] = false;

}

}

}

for (int i = 0; i < segs; i++)

if (check2[i] && !check3[i]) {

retseg = segmentSorter(str);

seg1 = Integer.parseInt(retseg[1][i]);

seg2 = Integer.parseInt(retseg[2][i]);

path1 = retseg[0][i].substring(1, retseg[0][i].length() - 1);

retterm = termSorter(path1);

terms = Integer.parseInt(retterm[3][0]);

for (int j = 0; j < terms; j++) {

term1 = Integer.parseInt(retterm[1][j]);

term2 = Integer.parseInt(retterm[2][j]);

path2 = retterm[0][j];

mod = "1";

mod2 = "";

retcon = constantSorter(path2);

cons = Integer.parseInt(retcon[3][0]);

check = new boolean[cons];

for (k = 0; k < cons; k++) {

con = Integer.parseInt(retcon[1][k]);

check[k] = false;

if (k > 0 && path2.charAt(con - 1) == '/')

check[k] = true;

}

for (k = 0; k < cons; k++)

if (check[k])

mod += "\*" + retcon[0][k];

else

mod2 += "\*" + retcon[0][k];

path2 = "{" + mod2.substring(1) + "}/{" + mod + "}";

if (j == 0)

mod3 = path2;

else

mod3 += path1.substring(termn + 1, term1) + path2;

termn = term2;

}

path1 = "(" + mod3 + ")";

ret = str.substring(0, seg1);

ret += path1;

if (seg2 != str.length() - 1)

ret += str.substring(seg2 + 1);

str = ret;

check3[i] = true;

}

}

ret = bracketHolder(ret);

return ret;

}

String termManager2(String str) {

String mod = "";

boolean flag = false;

String[][] retterm, retcon;

String mod2;

int term1, term2, termn = 0, terms, con, cons, divider, divide[], k;

boolean[] check;

retterm = termSorter(str);

terms = Integer.parseInt(retterm[3][0]);

String ret = "";

for (int i = 0; i < terms; i++) {

term1 = Integer.parseInt(retterm[1][i]);

term2 = Integer.parseInt(retterm[2][i]);

retcon = constantSorter(retterm[0][i]);

cons = Integer.parseInt(retcon[3][0]);

divider = cons - 1;

divide = new int[divider];

for (int j = 0; j < cons; j++) {

con = Integer.parseInt(retcon[2][j]);

if (j < divider)

divide[j] = con + 1;

}

flag = true;

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

if (retterm[0][i].charAt(divide[j]) != '/')

flag = false;

if (flag) {

flag = false;

for (int j = 0; j < retterm[0][i].length(); j++)

if (retterm[0][i].charAt(j) == '{')

flag = true;

if (flag) {

if (divider == 1)

mod = retterm[0][i];

if (divider == 2) {

if (retterm[0][i].charAt(divide[1] - 1) == '}')

mod = retcon[0][0] + "/{" + retcon[0][2] + "\*" + retcon[0][1] + "}";

else

mod = "{" + retcon[0][0] + "\*" + retcon[0][2] + "}/" + retcon[0][1];

}

if (divider == 3)

mod = "{" + retcon[0][0] + "\*" + retcon[0][3] + "}/{" + retcon[0][2] + "\*" + retcon[0][1] + "}";

} else

mod = retterm[0][i];

} else {

mod = "1";

mod2 = "";

check = new boolean[cons];

for (k = 0; k < cons; k++) {

con = Integer.parseInt(retcon[1][k]);

check[k] = false;

if (k > 0 && retterm[0][i].charAt(con - 1) == '/')

check[k] = true;

}

for (k = 0; k < cons; k++)

if (check[k])

mod += "\*" + retcon[0][k];

else

mod2 += "\*" + retcon[0][k];

mod = "{" + mod2.substring(1) + "}/{" + mod + "}";

}

if (i == 0)

ret = mod;

else

ret += str.substring(termn + 1, term1) + mod;

termn = term2;

}

return ret;

}

String termManager3(String str) {

String mod = "";

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

if (str.charAt(i) == '(')

mod += "{";

else if (str.charAt(i) == ')')

mod += "}";

else

mod += str.charAt(i);

str = mod;

return str;

}

void divisionHandler() {

String mod = "";

boolean flag = false;

String[][] retseg, retterm;

int seg1, seg2, segs, term1, term2, termn = 0, terms, k, capture = 0, loop;

String str1 = "", str2 = "", path1, path2, mod2 = "", retterm1[], retterm2[];

boolean check, check2[], check3[];

for (loop = 1; loop < 3; loop++) {

switch (loop) {

case 1:

str1 = former;

str2 = latter;

break;

case 2:

str1 = latter;

str2 = former;

break;

}

str1 = termManager(str1);

segs = Integer.parseInt(segmentSorter(str1)[3][0]);

check2 = new boolean[segs];

check3 = new boolean[segs];

for (int i = 0; i < segs; i++) {

check2[i] = false;

check3[i] = false;

}

while (true) {

flag = true;

for (int i = 0; i < segs; i++)

if (!check2[i])

flag = false;

if (flag)

break;

for (int i = 0; i < segs; i++) {

check2[i] = true;

mod = segmentSorter(str1)[0][i];

mod = mod.substring(1, mod.length() - 1);

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

if (mod.charAt(j) == '(') {

for (k = 0; k < segs; k++)

if ((j + 1 + Integer.parseInt(segmentSorter(str1)[1][i])) == Integer

.parseInt(segmentSorter(str1)[1][k]))

capture = k;

if (!check2[capture])

check2[i] = false;

}

}

}

for (int i = 0; i < segs; i++)

if (check2[i] && !check3[i]) {

retseg = segmentSorter(str1);

seg1 = Integer.parseInt(retseg[1][i]);

seg2 = Integer.parseInt(retseg[2][i]);

path1 = retseg[0][i].substring(1, retseg[0][i].length() - 1);

path1 = termManager2(path1);

retterm = termSorter(path1);

terms = Integer.parseInt(retterm[3][0]);

retterm1 = new String[terms];

retterm2 = new String[terms];

for (int j = 0; j < terms; j++) {

check = true;

path2 = retterm[0][j];

for (k = 0; k < path2.length(); k++)

if (path2.charAt(k) == '/')

check = false;

if (check) {

retterm1[j] = path2;

retterm2[j] = "1";

} else {

retterm1[j] = path2.substring(0, path2.indexOf('/'));

retterm2[j] = path2.substring(path2.indexOf('/') + 1);

}

}

mod = "1";

for (int j = 0; j < terms; j++) {

mod += "\*" + retterm2[j];

term1 = Integer.parseInt(retterm[1][j]);

term2 = Integer.parseInt(retterm[2][j]);

path2 = retterm1[j];

for (k = 0; k < terms; k++)

if (k != j)

path2 += "\*" + termManager3(retterm2[k]);

if (j == 0)

mod2 = path2;

else

mod2 += path1.substring(termn + 1, term1) + path2;

termn = term2;

}

path1 = "(" + mod2 + ")/{" + mod + "}";

mod = str1.substring(0, seg1);

mod += path1;

if (seg2 != str1.length() - 1)

mod += str1.substring(seg2 + 1);

str1 = mod;

check3[i] = true;

}

}

str1 = bracketHolder(str1);

mod = str1.substring(str1.indexOf('/') + 1);

str1 = str1.substring(0, str1.indexOf('/'));

switch (loop) {

case 1:

former = str1;

latter = "(" + str2 + ")\*" + mod;

break;

case 2:

latter = str1;

former = str2 + "\*" + mod;

break;

}

}

}

String bracketHandler(String str) {

String mod = "";

boolean flag = false;

String[][] retterm, retcon;

String retseg, path, ret = "";

int m = 0, n = str.length() - 1, brace, seg1, seg2, term1, term2, termn = 0, terms, con, cons;

while (Integer.parseInt(segmentSorter(str)[3][0]) != 0) {

retseg = segmentSorter(str)[0][0];

seg1 = Integer.parseInt(segmentSorter(str)[1][0]);

seg2 = Integer.parseInt(segmentSorter(str)[2][0]);

brace = 0;

for (int i = seg1; i >= 0; i--) {

if (str.charAt(i) == ')') {

brace--;

flag = false;

}

if (str.charAt(i) == '(') {

brace++;

if (brace == 1)

flag = true;

}

if (flag) {

if (str.charAt(i) == '+') {

m = (i + 1) \* 10;

break;

} else if (str.charAt(i) == '-') {

m = (i + 1) \* 10 + 1;

break;

} else

m = i \* 10 + 2;

}

}

brace = 0;

for (int i = seg2; i < str.length(); i++) {

if (str.charAt(i) == '(') {

brace++;

flag = false;

}

if (str.charAt(i) == ')') {

brace--;

if (brace == -1)

flag = true;

}

if (flag) {

if (str.charAt(i) == '+') {

n = (i - 1) \* 10;

break;

} else if (str.charAt(i) == '-') {

n = (i - 1) \* 10 + 1;

break;

} else

n = i \* 10 + 2;

}

}

if (n / 10 != str.length() - 1)

path = str.substring((m / 10), (n / 10 + 1));

else

path = str.substring(m / 10);

retterm = termSorter(retseg.substring(1, retseg.length() - 1));

terms = Integer.parseInt(retterm[3][0]);

retcon = constantSorter(path);

cons = Integer.parseInt(retcon[3][0]);

for (int i = 0; i < terms; i++) {

term1 = Integer.parseInt(retterm[1][i]);

term2 = Integer.parseInt(retterm[2][i]);

if (i == 0)

mod = retterm[0][i];

else {

if (m % 10 == 1) {

if ((retseg.substring(1, retseg.length() - 1)).substring(termn + 1, term1).equals("+"))

mod += '-';

else

mod += '+';

} else

mod += (retseg.substring(1, retseg.length() - 1)).substring(termn + 1, term1);

mod += retterm[0][i];

}

termn = term2;

for (int j = 0; j < cons; j++) {

con = Integer.parseInt(retcon[1][j]);

if (con != Integer.parseInt(segmentSorter(path)[1][0]))

mod += "\*" + retcon[0][j];

}

}

ret = str.substring(0, (m / 10));

ret += mod;

if ((n / 10) != str.length() - 1)

ret += str.substring((n / 10) + 1);

str = ret;

}

return ret;

}

String numeralManager(String str) {

String mod = "";

str = decimalHandler(str);

String[][] retterm, retcon, retnum;

int nums, cons, term1, term2, termn = 0, terms;

double n;

retterm = termSorter(str);

terms = Integer.parseInt(retterm[3][0]);

for (int i = 0; i < terms; i++) {

retcon = constantSorter(retterm[0][i]);

cons = Integer.parseInt(retcon[3][0]);

retnum = numeralSorter(retterm[0][i]);

nums = Integer.parseInt(retnum[3][0]);

mod = "";

n = 1.0;

for (int j = 0; j < cons; j++)

if (Character.isLetter(retcon[0][j].charAt(0)))

mod += "\*" + retcon[0][j];

for (int j = 0; j < nums; j++)

n \*= Double.parseDouble(retnum[0][j]);

retterm[0][i] = String.valueOf(n) + mod;

}

for (int i = 0; i < terms; i++) {

term1 = Integer.parseInt(retterm[1][i]);

term2 = Integer.parseInt(retterm[2][i]);

if (i == 0)

mod = retterm[0][i];

else

mod += str.substring(termn + 1, term1) + retterm[0][i];

termn = term2;

}

return mod;

}

void manager() {

String str;

String[][] retterm;

int term, terms;

char[] c;

boolean[] star, location;

str = former + '=' + latter;

retterm = termSorter(str);

terms = Integer.parseInt(retterm[3][0]);

c = new char[terms];

star = new boolean[terms];

location = new boolean[terms];

for (int i = 0; i < terms; i++) {

term = Integer.parseInt(retterm[1][i]);

if (i == 0)

c[i] = '+';

else

c[i] = str.charAt(term - 1);

if (c[i] == '=')

c[i] = '+';

if (term < str.indexOf('='))

location[i] = true;

else

location[i] = false;

star[i] = false;

for (int j = 0; j < retterm[0][i].length(); j++)

if (retterm[0][i].charAt(j) == '\*')

star[i] = true;

}

former = "";

latter = "";

for (int i = 0; i < terms; i++)

if (star[i]) {

if (!location[i]) {

if (c[i] == '+')

c[i] = '-';

else

c[i] = '+';

}

former += c[i] + retterm[0][i];

} else {

if (location[i]) {

if (c[i] == '+')

c[i] = '-';

else

c[i] = '+';

}

latter += c[i] + retterm[0][i];

}

str = former + '=' + latter;

String mod = "";

if (former.charAt(0) == '-') {

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

if (str.charAt(i) == '+')

mod += '-';

else if (str.charAt(i) == '-')

mod += '+';

else

mod += str.charAt(i);

str = mod;

}

str = str.substring(1);

former = str.substring(0, str.indexOf('='));

latter = "0" + str.substring(str.indexOf('=') + 1);

}

void combiner() {

String str = "";

for (int i = 0; i < 2; i++) {

switch (i) {

case 0:

str = former;

break;

case 1:

str = latter;

break;

}

String[][] retterm = termSorter(str), retcon;

int terms = Integer.parseInt(retterm[3][0]), cons;

double n = 0.0, current;

char c;

for (int j = 0; j < terms; j++) {

retcon = constantSorter(retterm[0][j]);

cons = Integer.parseInt(retcon[3][0]);

if (cons > 2)

endIt(7);

cons = Integer.parseInt(retterm[1][j]);

c = '+';

if (cons > 0)

c = str.charAt(cons - 1);

current = Double.parseDouble(retcon[0][0]);

if (c == '+')

n += current;

else

n -= current;

}

str = String.valueOf(n);

switch (i) {

case 0:

former = str + '\*' + x;

break;

case 1:

latter = str;

break;

}

}

}

void solver() {

double form = Double.parseDouble(former.substring(0, former.indexOf('\*'))), lat = Double.parseDouble(latter);

former = "";

former += x;

latter = String.valueOf(lat / form);

}

void finalizer() {

a = former + '=' + latter;

}

void doIt() {

check();

initializer();

former = symbolHolder(bracketHolder(former));

latter = symbolHolder(bracketHolder(latter));

divisionHandler();

former = numeralManager(bracketHandler(former));

latter = numeralManager(bracketHandler(latter));

manager();

combiner();

solver();

finalizer();

}

void endIt(int n) {

String end = "";

switch (n) {

case 1:

end = "invalid input";

break;

case 2:

end = "= missing";

break;

case 3:

end = "no var";

break;

case 4:

end = ">1 var";

break;

case 5:

end = "() order wrong";

break;

case 6:

end = "() not closed";

break;

case 7:

end = "x^a not allowed";

break;

}

System.out.print("\n\n\n\n\nError:\n" + end + "\nProram Failed.\n\n\n\n\n");

System.exit(0);

}

public LinearEquation(String eq) {

input(eq);

doIt();

System.out.println(a);

}

}

public class LETest {

public static void main(String[] args) {

String YOUREQUATION = "5/(t+4)=4";

new LinearEquation(YOUREQUATION);

}

}//Output: t=-2.75

public class LETest2 {

public static void main(String[] args) {

String YOUREQUATION = "5/(t+1)=3+(9-8)";

new LinearEquation(YOUREQUATION);

}

}//Output: t=0.25