Java API

String

|  |  |
| --- | --- |
| int | [**compareToIgnoreCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#compareToIgnoreCase(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) str)  Compares two strings lexicographically, ignoring case differences. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**concat**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#concat(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) str)  Concatenates the specified string to the end of this string. |
| boolean | [**contains**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#contains(java.lang.CharSequence))([**CharSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html) s)  Returns true if and only if this string contains the specified sequence of char values. |
| boolean | [**contentEquals**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#contentEquals(java.lang.CharSequence))([**CharSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html) cs)  Compares this string to the specified CharSequence. |
| boolean | [**contentEquals**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#contentEquals(java.lang.StringBuffer))([**StringBuffer**](https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuffer.html) sb)  Compares this string to the specified StringBuffer. |
| boolean | [**equalsIgnoreCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#equalsIgnoreCase(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) anotherString)  Compares this String to another String, ignoring case considerations. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**replace**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#replace(char,%20char))(char oldChar, char newChar)  Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**replace**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#replace(java.lang.CharSequence,%20java.lang.CharSequence))([**CharSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html) target, [**CharSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html) replacement)  Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**replaceAll**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#replaceAll(java.lang.String,%20java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) regex, [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) replacement)  Replaces each substring of this string that matches the given [**regular expression**](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum) with the given replacement. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**replaceFirst**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#replaceFirst(java.lang.String,%20java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) regex, [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) replacement)  Replaces the first substring of this string that matches the given [**regular expression**](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum) with the given replacement. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)[] | [**split**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#split(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) regex)  Splits this string around matches of the given [**regular expression**](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum). |
| boolean | [**startsWith**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#startsWith(java.lang.String,%20int))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) prefix, int toffset)  Tests if the substring of this string beginning at the specified index starts with the specified prefix. |
| [**CharSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html) | [**subSequence**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#subSequence(int,%20int))(int beginIndex, int endIndex)  Returns a new character sequence that is a subsequence of this sequence. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**substring**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#substring(int))(int beginIndex)  Returns a new string that is a substring of this string. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**substring**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#substring(int,%20int))(int beginIndex, int endIndex)  Returns a new string that is a substring of this string. |
| char[] | [**toCharArray**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toCharArray())()  Converts this string to a new character array. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toLowerCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toLowerCase())()  Converts all of the characters in this String to lower case using the rules of the default locale. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toLowerCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toLowerCase(java.util.Locale))([**Locale**](https://docs.oracle.com/javase/7/docs/api/java/util/Locale.html) locale)  Converts all of the characters in this String to lower case using the rules of the given Locale. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toString**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toString())()  This object (which is already a string!) is itself returned. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toUpperCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toUpperCase())()  Converts all of the characters in this String to upper case using the rules of the default locale. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toUpperCase**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#toUpperCase(java.util.Locale))([**Locale**](https://docs.oracle.com/javase/7/docs/api/java/util/Locale.html) locale)  Converts all of the characters in this String to upper case using the rules of the given Locale. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**trim**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#trim())()  Returns a copy of the string, with leading and trailing whitespace omitted. |

Arrays

|  |  |
| --- | --- |
| static void | [**fill**](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#fill(boolean%5B%5D,%20boolean))(boolean[] a, boolean val)  Assigns the specified boolean value to each element of the specified array of booleans. |
| static void | [**fill**](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#fill(boolean%5B%5D,%20int,%20int,%20boolean))(boolean[] a, int fromIndex, int toIndex, boolean val)  Assigns the specified boolean value to each element of the specified range of the specified array of booleans. |
| static void | [**sort**](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(byte%5B%5D))**(byte[] a)**  **Sorts the specified array into ascending numerical order.** |
| static void | [**sort**](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(byte%5B%5D,%20int,%20int))**(byte[] a, int fromIndex, int toIndex)**  **Sorts the specified range of the array into ascending order.** |
| static [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toString**](https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#toString(byte%5B%5D))**(byte[] a)**  **Returns a string representation of the contents of the specified array.** |

Math

|  |  |
| --- | --- |
| static double | [**abs**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#abs(double))(double a)  Returns the absolute value of a double value. |
| static double | [**acos**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#acos(double))**(double a)**  **Returns the arc cosine of a value; the returned angle is in the range 0.0 through pi.** |
| static double | [**asin**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#asin(double))**(double a)**  **Returns the arc sine of a value; the returned angle is in the range -pi/2 through pi/2.** |
| static double | [**atan**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#atan(double))**(double a)**  **Returns the arc tangent of a value; the returned angle is in the range -pi/2 through pi/2.** |
| static double | [**atan2**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#atan2(double,%20double))**(double y, double x)**  **Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).** |
| static double | [**ceil**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#ceil(double))**(double a)**  **Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.** |
| static double | [**floor**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#floor(double))**(double a)**  **Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.** |
| static double | [**max**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#max(double,%20double))**(double a, double b)**  **Returns the greater of two double values.** |
| static double | [**min**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#min(double,%20double))**(double a, double b)**  **Returns the smaller of two double values.** |
| static double | [**pow**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#pow(double,%20double))**(double a, double b)**  **Returns the value of the first argument raised to the power of the second argument.** |
| static double | [**random**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#random())**()**  **Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.** |
| static double | [**toDegrees**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#toDegrees(double))**(double angrad)**  **Converts an angle measured in radians to an approximately equivalent angle measured in degrees.** |
| static double | [**toRadians**](https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#toRadians(double))**(double angdeg)**  **Converts an angle measured in degrees to an approximately equivalent angle measured in radians.** |

gcd

public static long gcd(long a, long b)

{

if (b == 0)

return a;

return gcd(b, a%b);

}

balance the chemical reaction

import java.util.Scanner;

public class Balance

{

public static final int MAXELEM = 30;

public static String [] names = new String[MAXELEM];

public static int nElem = 0;

public static long [][] linsys = new long[MAXELEM][MAXELEM];

public static long gcd(long a, long b)

{

if (b == 0)

return a;

return gcd(b, a%b);

}

public static int getIndex(String ename)

{

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

if(names[i].equals(ename))

return i;

names[nElem++] = ename;

return nElem-1;

}

public static int processTerm(String t, int i, long [] coeffs)

{

while (i < t.length()) {

if (t.charAt(i) == ')') {

return i+1;

}

if (t.charAt(i) == '(') {

long [] tmpCoeffs = new long[MAXELEM];

i++;

i = processTerm(t, i, tmpCoeffs);

long factor = 0;

while (i<t.length() && Character.isDigit(t.charAt(i))) {

factor = 10\*factor + t.charAt(i) - '0';

i++;

}

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

coeffs[j] += tmpCoeffs[j]\*factor;

}

}

else {

String elem = "";

elem += t.charAt(i);

i++;

if (i<t.length() && Character.isLowerCase(t.charAt(i))) {

elem += t.charAt(i);

i++;

}

int index = getIndex(elem);

int val = 0;

while (i<t.length() && Character.isDigit(t.charAt(i))) {

val = 10\*val + t.charAt(i) - '0';

i++;

}

if (val > 0)

coeffs[index] = val;

else

coeffs[index] = 1;

}

}

return i;

}

public static boolean solve(long [][] a, int n, int m, long [] soln)

//

// solve linear system on n equations with m unknowns, checking

// for unsolvable systems

{

if (n < m)

return false;

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

// find non-zero value for a[i][i]

int j=i;

while (j<n && a[j][i] == 0)

j++;

if (j == n)

return false;

for(int k=0; k<m+1; k++) {

long tmp = a[i][k];

a[i][k] = a[j][k];

a[j][k] = tmp;

}

for(j=i+1; j<n; j++) {

if ((a[j][i]/a[i][i])\*a[i][i] != a[j][i]) {

for(int k=i; k<m+1; k++)

a[j][k] \*= a[i][i];

}

long mult = a[j][i]/a[i][i];

for(int k=i; k<m+1; k++) {

a[j][k] -= a[i][k]\*mult;

}

}

}

for(int i=m; i<n; i++) {

if (a[i][m] != 0)

return false;

}

// back substitution

for(int i=m-1; i>=0; i--) {

long val = a[i][m];

for(int j=i+1; j<m; j++)

val -= a[i][j]\*soln[j];

if ((val/a[i][i])\*a[i][i] != val) {

long factor = a[i][i]/gcd(val, a[i][i]);

for(int k=0; k<m+1; k++) {

soln[k] \*= factor;

a[k][m] \*= factor;

}

val \*= factor;

}

soln[i] = val/a[i][i];

}

return true;

}

public static boolean badSolution(long [] soln, int size)

{

if (soln[0] < 0) {

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

soln[i] \*= -1;

}

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

if (soln[i] <= 0)

return true;

}

return false;

}

public static void main(String [] args)

{

int r, p, icase=0;

long [] coeffs = new long[MAXELEM];

long [] soln = new long[MAXELEM];

Scanner in = new Scanner(System.in);

r = in.nextInt();

p = in.nextInt();

while (r > 0) {

icase++;

nElem = 0;

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

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

linsys[i][j] = 0;

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

String term;

term = in.next();

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

coeffs[j] = 0;

int index = 0;

index = processTerm(term, index, coeffs);

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

linsys[j][i] = coeffs[j];

}

}

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

String term;

term = in.next();

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

coeffs[j] = 0;

int index = 0;

index = processTerm(term, index, coeffs);

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

linsys[j][r+i] = -coeffs[j];

}

}

System.out.print("Case " + icase + ": ");

if (nElem < r+p-1)

System.out.println("No");

else {

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

linsys[j][r+p-1] \*= -1;

}

soln[r+p-1] = 1;

boolean solved = solve(linsys, nElem, r+p-1, soln);

if (!solved || badSolution(soln, r+p)) {

System.out.println("No");

}

else {

long factor = gcd(soln[0], soln[1]);

int i=2;

while (factor > 1 && i < r+p) {

factor = gcd(factor, soln[i]);

i++;

}

System.out.print(soln[0]/factor);

for(i=1; i<r+p; i++)

System.out.print(" " + soln[i]/factor);

System.out.println();

}

}

r = in.nextInt();

p = in.nextInt();

}

}

}