

# Problem Number 1: Polynomials

A polynomial of degree  $n$  has the following form:  $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  where  $a_0, a_1, \dots, a_n$  are numeric constants called the coefficients of the polynomial and  $a_n \neq 0$ . For example,  $5x^4 - 7x^3 + 3x + 1$  is a polynomial of degree 4 with integer coefficients. For purposes of this problem, all polynomials will have integer coefficients.

You are to create a polynomial package with operations to add, subtract, multiply, evaluate, and find the derivative of a single variable polynomial of degree  $n$ .

Commands are as follows:

evaluate $Z$	evaluate the polynomial on the next line at the value $Z$ .
add	followed by any number of polynomials to be summed, on successive lines.
subtract	followed by <i>polynomial</i> <sub>1</sub> and <i>polynomial</i> <sub>2</sub> on successive lines. The program will compute <i>polynomial</i> <sub>1</sub> - <i>polynomial</i> <sub>2</sub>
multiply	followed by any number of multiplicands on successive lines
derivative	followed by a line that contains a polynomial
last	A signal that the previous problem was the last one. The program terminates and no additional output is produced.

A blank line will always separate successive commands in the input. For the purposes of this problem you may assume that the input to your program will always be valid. No guarantees are given regarding spacing of the polynomial terms. The caret '^' will be used to represent exponentiation within a polynomial. The variable in the polynomial will always be  $x$  and the terms of the polynomial will always be given in decreasing order. If an  $a_i$  is equal to 1, the coefficient 1 may not be included in the data line (e.g.  $x^2+x+7$ ). The input to your program will always be lowercase.

The output from your program will include the example number followed by a colon and the answer. Spacing is not important. Coefficients that are 1 should not be shown (except in the case  $a_0=1$ ).

### **Sample Input**

evaluate 4  
 $3x^2+5$

add  
 $2x^3+7x^2+2x$   
 $4x^9+10x^2$   
 $x+2$

subtract  
 $5x^4 + 3x^2 + 5$   
 $2x^6 - 2x^2 + 3$

multiply  
 $2x + 1$   
 $3x^2 + x$   
 $x$

derivative  
 $5x^4+3x^3+2x+5$

last

### **Sample Output**

1: 53  
2:  $4x^9 + 2x^3+17x^2+3x+2$   
3:  $-2x^6+5x^4+5x^2+2$   
4:  $6x^4+5x^3+x^2$   
5:  $20x^3+9x^2+2$

## Problem Number 2: To Boldly Go

Space ships in the 25<sup>th</sup> century are driven by "warp" drive. Unfortunately, if two warp drives operate simultaneously within 10 light years of each other both space ships blow up leaving a really nasty mess that has to be cleaned up. You are to implement software for space traffic control (STC) to insure the safe operation of space ships using warp drive. All space ships must file a flight plan with STC. STC reviews each request in the order it was received and issues flight clearance to the space ship if its operation will not interfere with any previously cleared space ships. If the space ship interferes with another cleared space ship then clearance is denied and the space ship is not allowed to operate. A space ship that starts moving at the exact same instance when another space ship stops operation does not cause a problem.

Your program is to read a number  $n$ , the number of clearance requests followed by  $n$  clearance requests, one per line. A clearance request consists of a string space ship name followed by  $x_1, y_1, z_1, t_1$  coordinates of the origination point in space-time followed by  $x_2, y_2, z_2, t_2$  coordinates of the destination coordinates. All distances are in light years and time is given in minutes.  $t_2$  is always greater than  $t_1$ . The proposed flight plan starts at  $x_1, y_1, z_1$  at time  $t_1$  and moves at constant velocity to  $x_2, y_2, z_2$ , arriving at  $t_2$ . Your program is to consider the clearance requests in order and to print for each one the name of the space ship followed by the word "granted" if the flight will never come within 10 light years of any previously cleared flight or the word "denied" if the proposed flight plan will come within 10 light years of another operating space ship.

### Sample Input

```
2
ship1 0 0 0 0 15 0 0 1
ship2 0 15 0 0 0 0 0 1
```

### Sample Output

```
ship1 granted
ship2 granted
```

### Sample Input

```
2
ship1 0 0 0 0 14 0 0 1
ship2 0 14 0 0 0 0 0 1
```

### Sample Output

```
ship1 granted
ship2 denied
```

**Sample Input**

3

```
ship1 0 0 0 0 14 0 0 1  
ship2 0 14 0 0 0 0 0 1  
ship3 0 14 0 0 0 15 0 1
```

**Sample Output**

```
ship1 granted  
ship2 denied  
ship3 granted
```

**Sample Input**

2

```
ship1 0 0 0 0 0 0 0 1  
ship2 11 0 0 0 11 0 0 1
```

**Sample Output**

```
ship1 granted  
ship2 granted
```

## Problem Number 3: Fragment Reassembly

A biological computing system uses strings consisting of the letters 'A', 'G', 'C', and 'U' to represent numbers. The strings are converted to the corresponding number by processing the letters in the string from left to right in groups of three letters. The three letter groups in the string are translated to the digits in the number as shown in the table given below.

Triplet	Digit
GCA, GCC, GCG, GCU	1
UGC, UGU	2
GAC, GAU	3
GAA, CAG	4
GGA, GGC, GGG, GGU	5
AUG	6
AAC, AAU	7
AGA, AGG, CGA, CGC, CGG, CGU	8
UGG	9
UAA, UAG, UGA	<i>Stop</i>

The number of characters in an input string is always a multiple of three and will always end with 'UAA', 'UAG', or 'UGA'. For example, the string 'GCAUGUUGGUAA' translates to the string '129'. Note that there is not a one-to-one relationship between a number and the strings that encode it. For example, the strings 'GCAUGUUGGUAA', 'GCCUGCUGGUAG', and 'GCUUGCUGGUGA' all encode the number 129.

Researchers in the laboratory can isolate several copies of a string, and the number that it encodes, but because of the extraction process, each of the copies of the string is chopped up into an arbitrary number of fragments of arbitrary length. Not all of the copies are chopped up in the same way. The process guarantees that it will always be possible to reconstitute a complete string from the fragments it produces.

Write a program that will accept as input a number and an arbitrary number of fragments of copies of a string that encodes the number (the word 'end' signifies the end of the input). The program will produce as output a list of fragments required to reconstruct the string that encodes the number. Each line of the output will contain a fragment, and the index of the first and last character in the fragment that identifies the portion of the fragment to use to reconstruct the string. The indices are zero-based (i.e., the first letter in the fragment is at index 0). The fragments must be listed in the order they appear in the reconstituted string.

**Sample Input**

129  
GCUUGCUGGUGA  
end

**Sample Output**

GCUUGCUGGUGA 0 11

**Sample Input**

129  
GCUUGC  
GCUGGUGA  
GCU  
end

**Sample Output**

GCUUGC 0 6  
GCUGGUGA 3 7

**Sample Input**

129  
GCUUGC  
CUG  
CUG  
UG  
GU  
GA  
end

**Sample Output**

GCUUGC 0 5  
UG 0 1  
GU 0 1  
GA 0 1

## Problem Number 4: Primitive Roots

We say that integer  $x$ ,  $0 < x < p$ , is a primitive root modulo odd prime  $p$  if and only if the set  $\{ (x^i \bmod p) \mid 1 = i = p-1 \}$  is equal to  $\{ 1, \dots, p-1 \}$ . For example, the consecutive powers of 3 modulo 7 are 3, 2, 6, 4, 5, 1, and thus 3 is a primitive root modulo 7.

Write a program which given any odd prime  $3 < p < 65536$  outputs the number of primitive roots modulo  $p$ .

You may assume that the input  $p$  to your program is an odd prime number. The output of your program should consist of a single number that gives the number of primitive roots.

### Sample Input

23

### Sample Output

10

### Sample Input

31

### Sample Output

8

### Sample Input

79

### Sample Output

24

### Sample Input

107

### Sample Output

52

## Problem 5: Election Day

The United States Constitution requires that the number of representatives each state has in the House of Representatives must be proportional to the population of the various states. However, this causes a problem because you cannot have fractional representatives.

For example, assume there are only three states with populations of 12431, 9471, and 10282 and the House has only 10 representatives (the numbers are deliberately small). Then the first state should have 3.86 representatives, the second state 2.94 representatives, and the third state should have 3.19 representatives. In this case, the solution is obviously is to round these to 4, 3, and 3 respectively and fortunately that adds up to 10 representatives. However, rounding does not always give the correct number of representatives and it also has other undesirable properties.

One method that has been used is this: choose a number of representatives so that the sum of the absolute values of the deviations is as small as possible, subject to the Constitutional requirement that each state must have at least one representative. The deviation is the number of representatives given (integer) minus the number of representatives calculated by proportion (decimal). Deviations are calculated as accurately as necessary. The deviations for the data above would be 0.14, 0.06, and -0.19 to two decimal places. If two or more solutions give exactly the same sum of the absolute values of the deviations, either is acceptable.

The input to your program will consist of three lines. The first line will give the number of states, the second line specifies the number of representatives in the House, and the third line gives the populations of each state. The third line of input will consist of a series of numbers (state populations) separated by white space. The population for state 1 is given first, followed by the population for state 2, and so on up to the population for state  $n$  (where  $n$  is the number of states).

Populations will be no more than 32,767. The number of states will not exceed 50. If the number of representatives is less than the number of states, your program should print the message “The number of representatives is too small” and terminate without producing any additional output.

Your program will produce one line of output for each state that gives the state number followed by the number of representatives for that state. The states must be listed in numerical order.

**Sample Input**

3  
10  
12431 9471 10282

**Sample Output**

1 4  
2 3  
3 3

**Sample Input**

5  
3  
10 20 50 9 12

**Sample Output**

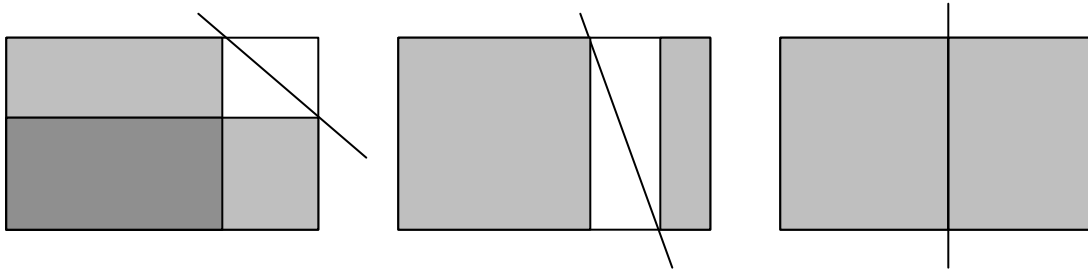
The number of representatives is too small

## Problem 6: Fighting City Hall

You are a proud owner of a rectangular piece of land in the city of Flatland. The edges of all pieces of property in Flatland are arranged to be horizontal and vertical in the coordinate system. The city identifies the boundaries of your property by specifying the coordinates of the top left and bottom right corners of your property. To make things simple the coordinates are always specified using positive integer values.

The city is in the process of expanding and has decided to build a number of roads. Roads in flatland are straight lines extending forever. The location of a road is given by specifying the coordinates of any two distinct points on the road. The city has published a list of the roads it wishes to build and will build the roads, one at a time, in the order they are listed.

If a road crosses your property the city will divide a portion of your property into two rectangles. The rectangles can overlap and they must share three edges with your current land as shown in the diagrams below:



After the town has subdivided your property into rectangles you will be allowed to retain ownership of one of the rectangles, but you must turn over the ownership of the rest of your property to the good citizens of Flatland. The process of dividing the property into rectangles will be repeated every time another road is built.

Write a program that accepts as input one line that contains four numbers that give the coordinates of a piece of property in flatland. This line will be followed by a series of lines that give the coordinates of the roads the town is planning to build, in the order they will be built. The last line of input will consist of four negative ones.

The program will produce as output the coordinates of the successive rectangles you should retain so that in the end you retain the maximum possible area.

**Sample Input**

0 0 5 5  
0 1 5 1  
-1 -1 -1 -1

**Sample Output**

0 1 5 5

## Problem 7: Oktoberfest in Outer Swabonia

Every year after all of the crops have been harvested the Swabian people celebrate the end of summer and the start of winter with Oktoberfest. The Swabian people love puzzles and have developed an interesting game that is used to distribute beer down a table. An arbitrary number of mugs of different sizes are lined up on a table. The first mug on the table is filled to capacity and all the other mugs are emptied. The goal of the game is to determine the shortest number of moves required to transfer a specified amount of beer from the first mug on the table to some other mug on the table (note that the destination mug may be the first mug).

During one move the contents of one mug is poured into another. You may pour some, or all of the beer, into another mug regardless of its size, however you must fill the mug. Therefore it is possible, and may be desirable, to overfill a mug resulting in a nasty spill. So, for example, pouring the entire contents of a mug that contains 3 units of beer into a mug that holds 2 units of beer will result in 1 unit of beer spilling on the floor. It is also possible to pour only 2 units of beer from the first mug into the second leaving 1 unit of beer in the first and 2 units of beer in the second. You are not allowed to pour only 1 unit of beer from the first mug into the second. Beer is poured in whole units only.

There are two variations to the game. In the first variation it is possible during a move to refill the first mug on the table. Note that a refill counts as a move. The second variation does not allow refills.

Write a program that will print out a shortest sequence of moves required to transfer the beer in the first mug on a table to some other mug. The first line of input will contain either the word “yes” or “no” to indicate whether or not refills are allowed. The second line will contain a series of numbers separated by white space. Each number gives the capacity of a mug on the table. The mugs are numbered starting at 1, so the first number gives the capacity of the first mug, the second number gives the capacity of the second mug and so on. The third line will contain two numbers. The first number specifies the mug the beer is to be transferred to and the second number specifies the amount of beer that should be in that mug at the end of the game.

The output of your program will consist of one line for each move in the game. The line will contain one number for each mug on the table. The number gives the amount of beer in the corresponding mug at the end of a turn. The letter ‘r’ will be printed at the end of a line to indicate if the first mug was refilled during that move. Taken together the lines will give a shortest sequence of moves required to move the beer from the first mug to the mug specified in the input. If it is impossible to move the beer as indicated, your program should simply print the message “not possible”.

**Sample Input**

no  
10 4 6  
3 2

**Sample Output**

10 0 0  
4 0 6  
4 4 2

**Sample Input**

yes  
2 7 4  
3 4

**Sample Output**

2 0 0  
0 0 2  
2 0 2 r  
0 0 4

**Sample Input**

no  
2 3 8  
3 4

**Sample Output**

not possible

## Problem 8: The Tower of Babble

The countries on the planet of Fozbot have decided to adopt the language Babble as the official language of the planet. The leaders of this planet have decided that in the year 2010 all Fozbotians will be required to speak only the language Babble.

Sentences in all languages spoken on the planet, including Babble, are written as a single line terminated by a new line character. A sentence consists of a sequence of words, where a word is defined as a sequence of lower case letters (a, b, c, ... z) delimited by one or more white space characters (spaces, tabs, or a new line character). Each country has been assigned the task of defining a set of rules that define how words in their native language can be translated to words in Babble. A translation rule can take one of two forms:

$$word_1 \rightarrow word_2$$
$$j[c_1c_2c_3\dots c_n] \rightarrow k[d_1d_2d_3\dots d_m]$$

where  $word_1$  and  $word_2$  are words in a language,  $c_i$  and  $d_i$  are characters in a language, and  $j$  and  $k$  are integers greater than zero.

The first rule specifies that any occurrences of  $word_1$  may be replaced by  $word_2$ . For example, the rule “gray -> grau” specifies that any occurrences of the word “gray” may be translated to the word “grau”. Note that the quotes are not part of the rule or the words they have been used to separate the rule from the text in this document.

The second rule states that  $j$  consecutive occurrences of letters  $c_1c_2c_3\dots c_n$  can be replaced by  $k$  consecutive occurrences of the letters  $d_1d_2d_3\dots d_m$ . For example, the rule “2[ab]->1[xyz]” would allow “ab”, “aa”, “bb”, or “ba” to be replaced by “x”, “y”, or “z”. It is valid for one set of characters, but not both, to be omitted. For example, “2[ab]->1[]” and “1[]->3[xyz]” are valid rules. The first rule states that “aa”, “ab”, “ba”, or “bb” may be replaced by nothing (i.e., they can be removed).

You are to write a program that accepts two sentences on separate lines followed by a series of translation rules (each on a separate line). The last translation rule will be followed by a line that contains a single ‘.’. Your program will determine if it is possible to translate the first sentence to the second using the rules given.

If translation is possible your program will print the first sentence on a single line followed by steps required to perform the translation. Each line will contain the number of the rule used, followed by a colon, followed by the result of applying the rule to the current string. The rules are numbered based on their order in the input. The first rule is rule 1, the next is rule 2, and so on. If a translation is not possible your program will print “No translation possible”.

**Sample Input**

ich bin da  
i am here  
ich -> i  
3[bin] -> 2[am]  
1[a] -> 3[here]  
1[d] -> 1[h]  
.

**Output Example:**

ich bin da  
1: i bin da  
3: i bin dere  
4: i bin here  
2: i am here

**Sample Input**

Elefanten sind grau  
it worked  
2[en] -> []  
Elefan -> I  
.

**Sample Output**

No translation possible